

Báo cáo Bài tập thực hành 1: Hệ thống Server Management System

Sinh viên: Nguyễn Minh Mạnh

Mục lục

I. Các công nghệ sử dụng	2
II. Mô tả hệ thống	2
III. Cơ sở dữ liệu	3
IV. Các chức năng	4
a) Kiểm tra trạng thái định kỳ của server:	4
b) Xác thực, phân quyền bằng JWT	5
c) Tạo Server	6
d) View server	8
e) Update server	14
f) Delete server	15
i) Import servers	16
j) Export servers	19
h) Báo cáo	20
V, Các yêu cầu phi chức năng	21
1. Sử dụng OpenAPI	21
2. Sử dụng Unit Test	22
2. Xác thực phân quyền bằng JWT	23
3. Redis Cache	23
4. Elasticsearch	24
5. Log và Logrotate	24
VI. Cách cài đặt:	26

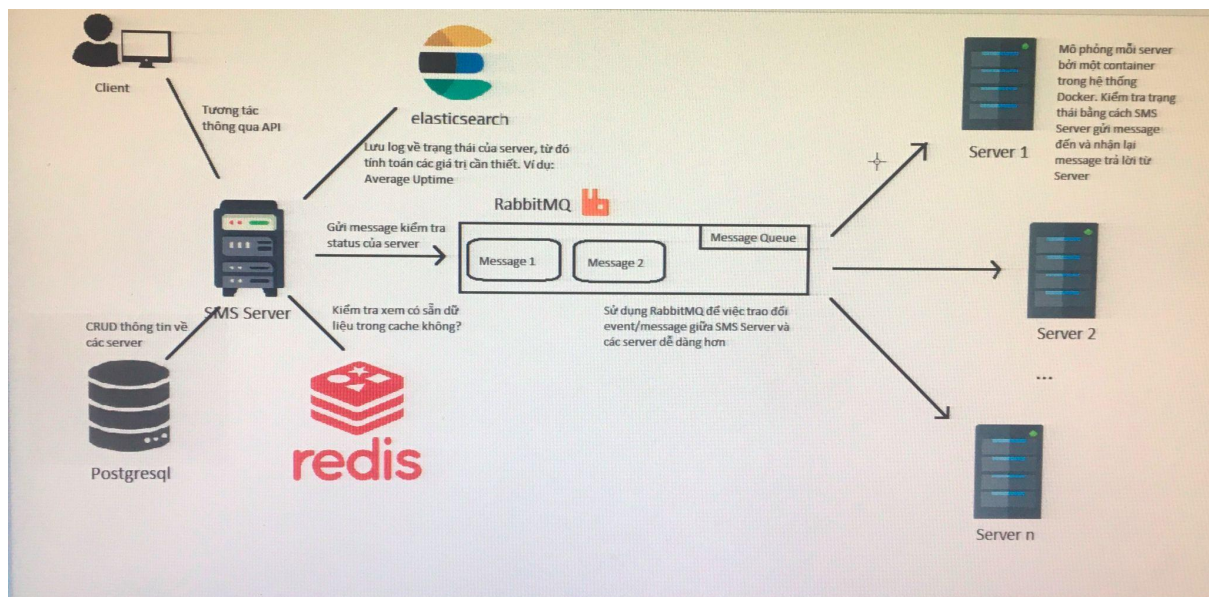
Link github: <https://github.com/minhmannh2001/vcs-sms>

Đề bài: Công ty VCS hiện có khoảng 1000 server. Xây dựng 1 hệ thống quản lý trạng thái On/Off của danh sách server này.

I. Các công nghệ sử dụng

- Ngôn ngữ: Golang, framework Gin
- Cơ sở dữ liệu: Postgresql, Elasticsearch
- Công cụ trực quan dữ liệu Kibana
- Message Queue: RabbitMQ
- Công nghệ đóng gói Docker
- Hệ thống cache Redis
- Công cụ xây dựng phát triển, thiết kế, xây dựng và làm tài liệu cho hệ thống OpenAPI - Swagger

II. Mô tả hệ thống



Hệ thống gồm các thành phần chính:

- SMS Server: Máy chủ quản lý các máy chủ khác
- Server n: máy chủ thứ n trong hệ thống
- Postgresql: Cơ sở dữ liệu
- Redis: Bộ đệm trong hệ thống
- RabbitMQ: Hàng đợi tin nhắn để trung chuyển tín hiệu kiểm tra trạng thái giữa các máy chủ
- Elasticsearch: Lưu trữ các tín hiệu phản hồi từ các máy chủ khác, dùng để trích xuất thông tin liên quan đến trạng thái của các máy chủ trong báo cáo

Các thành phần trên được đóng gói và chạy trên các docker container khác nhau.

Hiện tại trong hệ thống gồm 5 server sau:

Server Name	Container Name	IP
Server 1	sms-server1	172.22.0.8
Server 2	sms-server2	172.22.0.9
Server 3	sms-server3	172.22.0.10
Server 4	sms-server4	172.22.0.11
Server 5	sms-server5	172.22.0.12

III. Cơ sở dữ liệu

```
type Server struct {
    Id          int          `json:"id" gorm:"column:id;type:uuid;uuid_generate_v4();primary_key"`
    Name        string       `json:"name" gorm:"column:name;type:varchar(255);not null;uniqueIndex"`
    Ipv4        string       `json:"ipv4" gorm:"column:ipv4;type:varchar(15);not null;uniqueIndex"`
    User        string       `json:"user" gorm:"column:user;type:varchar(50)"`
    Password    string       `json:"password" gorm:"column:password;type:varchar(100)"`
    Status      string       `json:"status" gorm:"column:status;type:varchar(50)"`
    CreatedAt   *time.Time  `json:"created_at" gorm:"column:created_at;default:CURRENT_TIMESTAMP"`
    UpdatedAt   *time.Time  `json:"updated_at" gorm:"column:updated_at;default:CURRENT_TIMESTAMP;autoUpdateTime"`
}
```

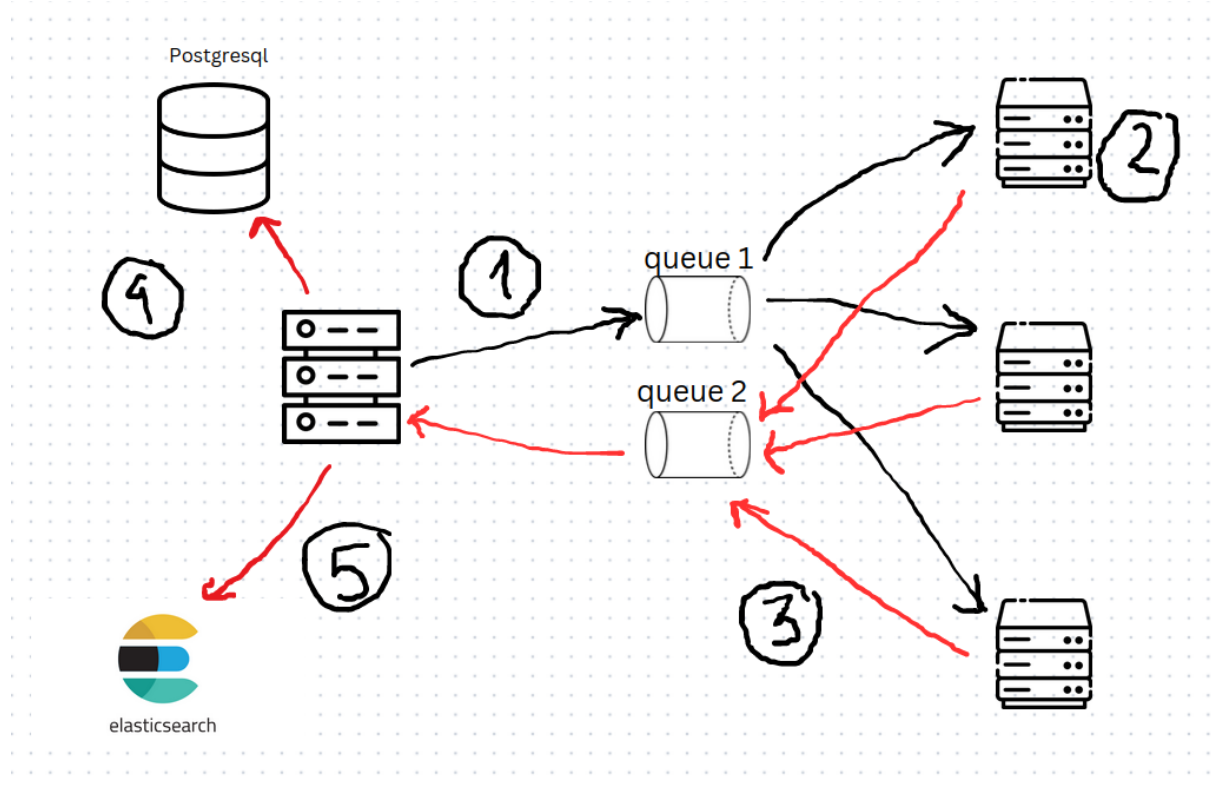
Định nghĩa struct Server với các trường thông tin và các mô tả cụ thể dùng để migrate sang cơ sở dữ liệu thông qua thư viện gorm

Column	Type	Collation	Nullable	Table "public.servers" Default	Storage	Compression	Stats target	Description
id	bigint		not null	nextval('servers_id_seq'::regclass)	plain			
name	character varying(255)		not null		extended			
ipv4	character varying(15)		not null		extended			
user	character varying(50)				extended			
password	character varying(100)				extended			
status	character varying(50)				extended			
created_at	timestamp with time zone			CURRENT_TIMESTAMP	plain			
updated_at	timestamp with time zone			CURRENT_TIMESTAMP	plain			
Indexes:								
"servers_pkey" PRIMARY KEY, btree (id)								
"idx_servers_ipv4" UNIQUE, btree (ipv4)								
"idx_servers_name" UNIQUE, btree (name)								
Access method: heap								

Bản mô tả dữ liệu trong CSDL Postgresql

IV. Các chức năng

a) Kiểm tra trạng thái định kỳ của server:



Cứ định kỳ 1 phút, SMS Server sẽ gửi 1 tín hiệu để xác định trạng thái của các server khác. Bao gồm các bước sau:

Bước 1: SMS Server gửi tín hiệu tới queue 1, từ đây sẽ được broadcast tới các server khác.

```
sms-sms-1 | 2023/03/15 03:28:16 [x] Sent sms:ping
```

Bước 2: Các server nhận tín hiệu từ queue 1, chuẩn bị gói tin để gửi lại cho SMS Server

Bước 3: Các server khác nếu có kết nối (trạng thái Up), sẽ gửi lại tín hiệu xác nhận tới queue 2

```
sms-server1-1 | 2023/03/15 03:28:16 172.22.0.11:pong
```

Bước 4: SMS server nhận tín hiệu từ queue 2, cập nhật trạng thái của từng server vào trong CSDL postgres

```
sms-sms-1 | 2023/03/15 03:28:16 Waiting for messages...
sms-sms-1 | 2023/03/15 03:28:16 Received a message: 172.22.0.11:pong
sms-sms-1 | 2023/03/15 03:28:16 ip: 172.22.0.11
```

Bước 5: SMS sau đó gửi các tín hiệu xác nhận này vào elastic DB, dùng để tổng hợp làm báo cáo định kỳ

Ta thấy rằng, chỉ có server2 và server4 có trạng thái Up, do SMS Server có kết nối đến các server này, còn các server1 và server3 có địa chỉ ip không tồn tại trong hệ thống mạng của docker.

```

sms=# SELECT * FROM servers;
 id | name | ipv4 | user | password | status | created_at | updated_at
-----+-----+-----+-----+-----+-----+-----+-----
  1 | server1 | 172.22.0.7 | admin | passw0rd | Down | 2023-03-13 10:55:40.5833+00 | 2023-03-15 04:38:17.303816+00
  3 | server3 | 172.22.0.13 | root | 123456a@ | Down | 2023-03-14 10:38:33.524271+00 | 2023-03-15 04:38:17.309779+00
  2 | server2 | 172.22.0.11 | admin | passw0rd | Up | 2023-03-13 17:13:18.669165+00 | 2023-03-15 04:38:17.312984+00
 14 | server4 | 172.22.0.10 | admin | passw0rd | Up | 2023-03-15 04:37:45.142965+00 | 2023-03-15 04:38:17.316978+00
(4 rows)

```

b) Xác thực, phân quyền bằng JWT

POST /login Login Handler

API authentication and authorization.

Parameters
Try it out

Name	Description
credential * required object (body)	Add credential Example Value Model <pre>{ "password": "string", "username": "string" }</pre> Parameter content type application/json

Responses
Response content type application/json

Code	Description
200	OK Example Value Model <pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</pre>
400	Bad Request Example Value Model <pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</pre>

Hệ thống sử dụng cơ chế xác thực JWT, ta cần đăng nhập để lấy token trước khi thực hiện các chức năng khác trong hệ thống.

Demo

1. URL để xác thực

POST
http://localhost:8080/login

2. Body chứa json gồm username và password

```

1 {
2   ... "username": "minhmannh2001",
3   ... "password": "123456aA@"
4 }

```

3. Sau khi gửi request, ta nhận được response chứa mã token

c) Tạo Server

Mô tả: Cho phép người dùng tạo 1 server với đầy đủ thông tin
Input: thông tin server dạng json trong request body
Output: Kết quả tạo server

Demo:

① Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative enviro

Token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1  
YW11IjoibWUaG1hbm50MjAwMSIsImFkbWUl  
pCncVClJleHA0IjE2Ng5Mc2N2DgsilmhkCl  
6MTY3ODg0IjE2OTQwIiwiaWF0IjoiZ21uZjcy  
b20ifQ.psuLQA4EQcncKkAhP9kg0INvJeQE  
Z7mYU7RozK1m5UJ
```

2. URL để gửi request

POST

http://localhost:8080/api/server

3. Nội dung phần body chứa thông tin về server sẽ thêm

```
1 {
2   "name": "server3",
3   "ipv4": "172.22.0.12",
4   "user": "root",
5   "password": "123456a@"
6 }
```

4. Nhận được response chứa message thông báo kết quả

- Trường hợp lỗi: vì trong hệ thống đã có chứa server3 nên ta đổi tên thành server 4

```
1 {
2   "status": false,
3   "message": "Cannot create new server with this information",
4   "token": "",
5   "errors": [
6     "duplicated key not allowed"
7   ],
8   "data": {}
9 }
```

- Trường hợp thành công:

```
1 {
2   "status": true,
3   "message": "Created successfully new server!",
4   "token": "",
5   "errors": null,
6   "data": {}
7 }
```

5. Kiểm tra trong cơ sở dữ liệu, ta thấy server 4 đã được thêm vào

id	name	ipv4	user	password	status	created_at	updated_at
1	server1	172.22.0.7	admin	passwd	Down	2023-03-13 10:55:40.5833+00	2023-03-15 04:18:17.00471+00
3	server3	172.22.0.13	root	123456a@	Down	2023-03-14 10:38:33.524271+00	2023-03-15 04:18:17.010442+00
2	server2	172.22.0.11	admin	passwd	Down	2023-03-13 17:13:18.669165+00	2023-03-15 04:18:17.012124+00
11	server4	172.22.0.12	root	123456a@	Up	2023-03-15 04:17:29.29861+00	2023-03-15 04:18:17.023181+00

(4 rows)

d) View server

GET

/api/servers

Export servers

⌵ 🔒

View or export servers based on url query

Parameters

Cancel

Name	Description
from	From
integer (query)	<input type="text" value="from"/>
to	To
integer (query)	<input type="text" value="to"/>
perpage	Account Per Page
integer (query)	<input type="text" value="perpage"/>
sortby	Sort By
string (query)	<input type="text" value="sortby"/>
order	Order
string (query)	<input type="text" value="order"/>
filter	Filter
string (query)	<input type="text" value="filter"/>
export	Export
string (query)	<input type="text" value="true"/>

Execute

Clear

Responses

Response content type application/json

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/servers?export=true' \
  -H 'accept: application/json' \
  -H 'Authorization: BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ3aW1IjoibWluaG1hbm50MjAwMSIsImFkbWwluIjp0cnVLLCJleHAiOjE2Nzg5MTM4NDU5MTU0OTY3ODgyNzQ0MSwiaXNzIjoic21zLnZjcysjb201f' \
  >/dev/null
```

Request URL

```
http://localhost:8080/api/servers?export=true
```

Server response

Code	Details
200	<div>Response body</div> <div>Download file</div> <div>Response headers</div> <div><pre>content-disposition: File Transfer content-disposition: attachment; filename=servers-20230314205741.xlsx content-type: application/octet-stream date: Tue, 14 Mar 2023 20:57:41 GMT transfer-encoding: chunked</pre></div>

Responses

Code	Description
200	<div>OK</div> <div>Example Value Model</div> <div><pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</pre></div>
400	<div>Bad Request</div> <div>Example Value Model</div> <div><pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</pre></div>

Mô tả: Lấy ra danh sách server, có thể kèm filter nếu người dùng nhập vào. Danh sách được phân trang. Có sort theo trường nào đó.

Input:

- Filter (optional)
- From, to (optional): thông tin phân trang

- sort, order (optional): trường và thứ tự sort

Output: Danh sách server phù hợp

Demo:

1. Thêm token JWT vào request

2. Phương thức và url để gửi request, có chứa các url params để điều chỉnh kết quả

- Nếu không chứa param, sẽ lấy ra tất cả server

```
GET http://localhost:8080/api/servers
```

- filter theo trạng thái up hoặc down

```
GET http://localhost:8080/api/servers?filter=Up
```

```
GET http://localhost:8080/api/servers?filter=Down
```

- filter theo trang, từ trang bao nhiêu đến trang bao nhiêu và số lượng kết quả mỗi trang

```
GET http://localhost:8080/api/servers?from=1&to=2&perpage=1
```

- sort theo trường và thứ tự sort

```
GET http://localhost:8080/api/servers?filter=Down&sort=created_at&order=asc
```

theo trường created_at, thứ tự tăng dần

```
GET http://localhost:8080/api/servers?filter=Down&sort=created_at&order=desc
```

theo trường created_at, thứ tự giảm dần

3. Kết quả trả về

- Trường hợp không có params, trả về tất cả

```

1  {
2    "status": true,
3    "message": "exporting servers...",
4    "token": "",
5    "errors": null,
6    "data": [
7      {
8        "id": 1,
9        "name": "server1",
10       "ipv4": "172.22.0.7",
11       "user": "admin",
12       "password": "passw0rd",
13       "status": "Down",
14       "created_at": "2023-03-13T10:55:40.5833Z",
15       "updated_at": "2023-03-15T04:48:17.493089Z"
16     },
17     {
18       "id": 3,
19       "name": "server3",
20       "ipv4": "172.22.0.13",
21       "user": "root",
22       "password": "123456a@",
23       "status": "Down",
24       "created_at": "2023-03-14T10:38:33.524271Z",
25       "updated_at": "2023-03-15T04:48:17.499108Z"
26     },
27     {
28       "id": 14,
29       "name": "server4",
30       "ipv4": "172.22.0.10",
31       "user": "admin",
32       "password": "passw0rd",
33       "status": "Up",
34       "created_at": "2023-03-15T04:37:45.142965Z",
35       "updated_at": "2023-03-15T04:48:17.502301Z"
36     },

```

- Trường hợp filter theo trạng thái down

```

1  {
2    "status": true,
3    "message": "exporting servers...",
4    "token": "",
5    "errors": null,
6    "data": [
7      {
8        "id": 1,
9        "name": "server1",
10       "ipv4": "172.22.0.7",
11       "user": "admin",
12       "password": "passw0rd",
13       "status": "Down",
14       "created_at": "2023-03-13T10:55:40.5833Z",
15       "updated_at": "2023-03-15T04:49:17.506212Z"
16     },
17     {
18       "id": 3,
19       "name": "server3",
20       "ipv4": "172.22.0.13",
21       "user": "root",
22       "password": "123456a@",
23       "status": "Down",
24       "created_at": "2023-03-14T10:38:33.524271Z",
25       "updated_at": "2023-03-15T04:49:17.512164Z"
26     },
27     {
28       "id": 14,
29       "name": "server4",
30       "ipv4": "172.22.0.10",
31       "user": "admin",
32       "password": "passw0rd",
33       "status": "Down",
34       "created_at": "2023-03-15T04:37:45.142965Z",
35       "updated_at": "2023-03-15T04:49:17.528631Z"
36     }
37   ]
38 }

```

- Trường hợp filter theo trạng thái Up

```

1  {
2      "status": true,
3      "message": "exporting servers...",
4      "token": "",
5      "errors": null,
6      "data": [
7          {
8              "id": 2,
9              "name": "server2",
10             "ipv4": "172.22.0.11",
11             "user": "admin",
12             "password": "passw0rd",
13             "status": "Up",
14             "created_at": "2023-03-13T17:13:18.669165Z",
15             "updated_at": "2023-03-15T04:49:17.545165Z"
16         }
17     ]
18 }

```

- filter từ trang 1 đến trang 2, mỗi trang 1 server

```

    "data": [
        {
            "id": 1,
            "name": "server1",
            "ipv4": "172.22.0.7",
            "user": "admin",
            "password": "passw0rd",
            "status": "Down",
            "created_at": "2023-03-13T10:55:40.5833Z",
            "updated_at": "2023-03-15T04:54:17.617434Z"
        },
        {
            "id": 3,
            "name": "server3",
            "ipv4": "172.22.0.13",
            "user": "root",
            "password": "123456a@",
            "status": "Down",
            "created_at": "2023-03-14T10:38:33.524271Z",
            "updated_at": "2023-03-15T04:54:17.62147Z"
        }
    ]

```

- filter theo trạng thái down, xếp theo trường created_at theo chiều tăng dần

```
"data": [  
  {  
    "id": 1,  
    "name": "server1",  
    "ipv4": "172.22.0.7",  
    "user": "admin",  
    "password": "passwd",  
    "status": "Down",  
    "created_at": "2023-03-13T10:55:40.5833Z",  
    "updated_at": "2023-03-15T04:53:17.595414Z"  
  },  
  {  
    "id": 3,  
    "name": "server3",  
    "ipv4": "172.22.0.13",  
    "user": "root",  
    "password": "123456a@",  
    "status": "Down",  
    "created_at": "2023-03-14T10:38:33.524271Z",  
    "updated_at": "2023-03-15T04:53:17.599764Z"  
  },  
  {  
    "id": 14,  
    "name": "server4",  
    "ipv4": "172.22.0.10",  
    "user": "admin",  
    "password": "passwd",  
    "status": "Down",  
    "created_at": "2023-03-15T04:37:45.142965Z",  
    "updated_at": "2023-03-15T04:53:17.601627Z"  
  }  
]
```

e) Update server

PUT /api/server/{id} Update server

Update server with provided information

Parameters Try it out

Name	Description
server * required object (body)	Update server <div>Example Value Model</div> <pre>{ "created_at": "string", "id": 0, "ipv4": "string", "name": "string", "password": "string", "status": "string", "updated_at": "string", "user": "string"}</pre> <div>Parameter content type application/json</div>
id * required integer (path)	Server ID <div>id</div>

Responses Response content type application/json

Code	Description
200	OK <div>Example Value Model</div> <pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string"}</pre>
400	Bad Request <div>Example Value Model</div> <pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string"}</pre>

Mô tả: cập nhật thông tin 1 server. Không cho phép cập nhật trường server_id

Input:

- server_id: server cần cập nhật
- update_data: các thông tin cần update cho server

Output: Thông tin kết quả cập nhật

Demo:

1. Thêm token JWT vào request
2. Phương thức và url để gửi request, thêm id của server cần update vào url

PUT ⌵ http://localhost:8080/api/server/1

3. Nội dung body chứa thông tin cần chỉnh sửa

```
1 {  
2   ... "name": "server4 will be deleted"  
3 }
```

4. Kết quả request được trả về trong response

```

1  {
2      "status": true,
3      "message": "Updated server successfully!",
4      "token": "",
5      "errors": null,
6      "data": {}
7  }

```

5. Kiểm tra lại trong CSDL, ta thấy tên của server đã thay đổi, ngoài ra trường updated_at cũng thay đổi

```

sms=# SELECT * FROM servers;
 id |      name      | ipv4   | user  | password | status |      created_at      |      updated_at
-----+-----+-----+-----+-----+-----+-----+-----
  1 | server1        | 172.22.0.7 | admin | passw0rd | Down   | 2023-03-13 10:55:40.5833+00 | 2023-03-15 04:26:17.144432+00
  3 | server3        | 172.22.0.13 | root  | 123456a@ | Down   | 2023-03-14 10:38:33.524271+00 | 2023-03-15 04:26:17.146639+00
  2 | server2        | 172.22.0.11 | admin | passw0rd | Up     | 2023-03-13 17:13:18.669165+00 | 2023-03-15 04:26:17.157692+00
11 | server4 will be deleted | 172.22.0.12 | root  | 123456a@ | Down   | 2023-03-15 04:17:29.29861+00 | 2023-03-15 04:26:40.791108+00
(4 rows)

```

f) Delete server

DELETE /api/server/{id} Delete server by ID

Using ID to delete server

Parameters
Try it out

Name	Description
id * required integer (path)	Server ID <input type="text" value="id"/>

Responses
Response content type: application/json

Code	Description
200	OK <div> Example Value Model </div> <pre> { "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" } </pre>
400	Bad Request <div> Example Value Model </div> <pre> { "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" } </pre>

Demo:

1. Thêm token JWT vào request
2. Phương thức và url để gửi request, ta thêm id của server cần xóa vào url

DELETE

3. Kết quả trả về thông qua response

```

1  {
2      "status": true,
3      "message": "Deleted server successfully!",
4      "token": "",
5      "errors": null,
6      "data": {}
7  }

```

4. Kiểm tra lại CSDL, thấy server 4 đã bị xóa

```

sms=# SELECT * FROM servers;
 id | name | ipv4 | user | password | status | created_at | updated_at
-----+-----+-----+-----+-----+-----+-----+-----
  1 | server1 | 172.22.0.7 | admin | passw0rd | Down | 2023-03-13 10:55:40.5833+00 | 2023-03-15 04:28:17.186207+00
  3 | server3 | 172.22.0.13 | root | 123456a@ | Down | 2023-03-14 10:38:33.524271+00 | 2023-03-15 04:28:17.189793+00
  2 | server2 | 172.22.0.11 | admin | passw0rd | Up | 2023-03-13 17:13:18.669165+00 | 2023-03-15 04:28:17.193751+00
(3 rows)

```

i) Import servers

POST /api/servers Import servers

Import servers within a excel file

Parameters
Try it out

Name	Description
server * <small>required</small> file <small>(formData)</small>	Update server <input type="button" value="Browse..."/> No file selected.

Responses
Response content type: application/json

Code	Description
200	OK Example Value Model <pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</pre>
400	Bad Request Example Value Model <pre>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</pre>

Mô tả: cho phép tạo 1 danh sách nhiều server từ file excel. Bỏ qua các server_id hoặc user_name đã tồn tại

Input: File cần import

Output: Danh sách server đã import thành công, danh sách server import thất bại

Demo:

1. Thêm token JWT vào request
2. Chuẩn bị file để import

	A	B	C	D	
	name	ipv4	user	password	
	server3	172.22.0.9	admin	123456aA@	
	server4	172.19.0.10	admin	passw0rd	
	Server 5	172.19.0.100	foo	bar	

Nội dung:

- server3 bị trùng tên
- server 4 hợp lệ
- server 5 không thể kết nối tới, không tồn tại địa chỉ ip

3. Phương thức và url để gửi request

POST

▼

http://localhost:8080/api/servers/

4. Chọn tệp để gửi lên, tệp nằm trong form-file của body

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	file	servers-to-import.xlsx ×	
	Key	Value	Description

5. Kết quả gửi về qua response, bao gồm danh sách server thêm thành công và danh sách server không thể thêm vào CSDL

```

"token": "",
"errors": null,
"data": {
  "created_servers": [
    {
      "id": 14,
      "name": "server4",
      "ipv4": "172.22.0.10",
      "user": "admin",
      "password": "passw0rd",
      "status": "Up",
      "created_at": "2023-03-15T04:37:45.142965Z",
      "updated_at": "2023-03-15T04:37:45.142965Z"
    }
  ],
  "uncreated_servers": [
    {
      "id": 0,
      "name": "server3",
      "ipv4": "172.22.0.9",
      "user": "admin",
      "password": "123456aA@",
      "status": "Up",
      "created_at": null,
      "updated_at": null
    },
    {
      "id": 0,
      "name": "Server 5",
      "ipv4": "172.19.0.100",
      "user": "foo",
      "password": "bar",
      "status": "",
      "created_at": null,
      "updated_at": null
    }
  ]
}

```

6. Kiểm tra lại CSDL, ta thấy server4 đã được thêm

```

sms=# SELECT * FROM servers;
 id | name   | ipv4   | user  | password | status |      created_at      |      updated_at
-----+-----+-----+-----+-----+-----+-----+-----
  1 | server1 | 172.22.0.7 | admin | passw0rd | Down   | 2023-03-13 10:55:40.5833+00 | 2023-03-15 04:38:17.303816+00
  3 | server3 | 172.22.0.13 | root  | 123456a@ | Down   | 2023-03-14 10:38:33.524271+00 | 2023-03-15 04:38:17.309779+00
  2 | server2 | 172.22.0.11 | admin | passw0rd | Up     | 2023-03-13 17:13:18.669165+00 | 2023-03-15 04:38:17.312984+00
 14 | server4 | 172.22.0.10 | admin | passw0rd | Up     | 2023-03-15 04:37:45.142965+00 | 2023-03-15 04:38:17.316978+00
(4 rows)

```

j) Export servers

GET

/api/serversExport servers

View or export servers based on url query

Parameters

Cancel

Name	Description
from integer <small>(query)</small>	From <input type="text" value="from"/>
to integer <small>(query)</small>	To <input type="text" value="to"/>
perpage integer <small>(query)</small>	Account Per Page <input type="text" value="perpage"/>
sortby string <small>(query)</small>	Sort By <input type="text" value="sortby"/>
order string <small>(query)</small>	Order <input type="text" value="order"/>
filter string <small>(query)</small>	Filter <input type="text" value="filter"/>
export string <small>(query)</small>	Export <input type="text" value="true"/>

Execute

Clear

Responses

Response content typeapplication/json

Curl

```
curl -X 'GET' \
'http://localhost:8080/api/servers?export=true' \
-H 'accept: application/json' \
-H 'Authorization: BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ2YWllIjoibWluaGlhbm5oMjAwMSIsImFkbWlucyIjp0cnVLCjleHAiojE2Zng5MTM4NDkxLmldCmY3ODgyNzQ0MSwiaXNzIjoic2lzLnZjcycyb20if' 
```

Request URL

```
http://localhost:8080/api/servers?export=true
```

Server response

Code	Details
200	<div>Response body Download file</div> <div>Response headers content-description: File Transfer content-disposition: attachment; filename=servers-20230314205741.xlsx content-type: application/octet-stream date: Tue, 14 Mar 2023 20:57:41 GMT transfer-encoding: chunked</div>

Responses

Code	Description
200	OK <div>Example Value Model</div> <div>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</div>
400	Bad Request <div>Example Value Model</div> <div>{ "data": "string", "errors": "string", "message": "string", "status": true, "token": "string" }</div>

Mô tả: Cho phép export 1 danh sách ra file excel

Demo:

1. Thêm token JWT vào request
2. url và phương thức gửi request, chứa param ?export=true

Parameters

Cancel

Name	Description
from integer (query)	From <input type="text" value="from"/>
to integer (query)	To <input type="text" value="to"/>
perpage integer (query)	Account Per Page <input type="text" value="perpage"/>
sortby string (query)	Sort By <input type="text" value="sortby"/>
order string (query)	Order <input type="text" value="order"/>
filter string (query)	Filter <input type="text" value="filter"/>
export string (query)	Export <input type="text" value="true"/>

Execute

Clear

3. Kết quả trả về

Code

Details

200

Response body

[Download file](#)

Response headers

content-description: File Transfer
content-disposition: attachment; filename=servers-20230315050519.xlsx
content-type: application/octet-stream
date: Wed, 15 Mar 2023 05:05:19 GMT
transfer-encoding: chunked

4 Mở file ra kiểm tra, file chứa thông tin về các server

A	B	C	D	E	F	G	H
ID	Server Name	IPV4	User	Password	Status	Created At	Updated At
1	server1	172.22.0.1	admin	passwd0rd	Down	2023-03-13 10:55:40.5833 +0000 UTC	2023-03-15 05:05:17.793049 +0000 UTC
3	server3	172.22.0.3	root	123456a@	Down	2023-03-14 10:38:33.524271 +0000 UTC	2023-03-15 05:05:17.796918 +0000 UTC
14	server4	172.22.0.14	admin	passwd0rd	Down	2023-03-15 04:37:45.142965 +0000 UTC	2023-03-15 05:05:17.81187 +0000 UTC
2	server2	172.22.0.2	admin	passwd0rd	Up	2023-03-13 17:13:18.669165 +0000 UTC	2023-03-15 05:05:17.826155 +0000 UTC

h) Báo cáo

GET

/api/servers/report

Report server information intentionally

⌵ 🔒

Report server information

Parameters

Try it out

No parameters

Responses

Response content type

application/json

⌵

Code

Description

200

OK

Example Value

Model

{
 "data": "string",
 "errors": "string",
 "message": "string",
 "status": true,
 "token": "string"
}

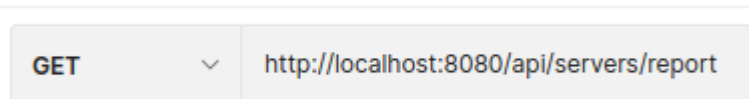
Định kỳ 30 phút thì gửi thông tin email cho người quản trị với thông báo trạng thái server trong vòng 3 tiếng vừa rồi:

- Các server trong hệ thống
- Số lượng server on
- Số lượng server off
- Tỷ lệ thời gian Uptime của server
- Các khoảng thời gian downtime của server

Xây dựng API để có thể chủ động report thông tin trên

Demo:

1. Thêm token JWT vào request
2. Phương thức và url để gửi request



3. Kết quả trả về thông qua response

```

1 {
2   "status": true,
3   "message": "Sending server report successfully",
4   "token": "",
5   "errors": null,
6   "data": {}
7 }

```

4. Kiểm tra email để xem thư báo cáo

The screenshot shows an email interface with a report titled "Report Server Status - Hello Admin! This is the report email." Below the email header is a table with server status data.

ID	Server Name	IPV4	User	Password	Status	Created At	Updated At	Percentage Of Uptime	Downtime Intervals
1	server1	172.22.0.7	admin	passwd0rd	Down	2023-03-13 10:55:40.5833 +0000 UTC	2023-03-15 05:10:17.879133 +0000 UTC	0.00%	{(from:2023-03-15 02:10:37.957266609 +0000 UTC
3	server3	172.22.0.13	root	123456a@	Down	2023-03-14 10:38:33.524271 +0000 UTC	2023-03-15 05:10:17.883567 +0000 UTC	0.00%	{(from:2023-03-15 02:10:37.969092102 +0000 UTC
14	server4	172.22.0.10	admin	passwd0rd	Up	2023-03-15 04:37:45.142965 +0000 UTC	2023-03-15 05:10:17.898051 +0000 UTC	100.00%	{}
2	server2	172.22.0.11	admin	passwd0rd	Up	2023-03-13 17:13:18.669165 +0000 UTC	2023-03-15 05:10:17.915493 +0000 UTC	55.87%	{(from:2023-03-15 02:10:37.98226455 +0000 UTC

Ta thấy rằng server 4 từ lúc khởi tạo đến hiện tại chưa bị tắt nên trạng thái Uptime là 100%

V, Các yêu cầu phi chức năng

1. Sử dụng OpenAPI

Tuân thủ và sử dụng Swagger để tạo tài liệu OpenAPI cho từng endpoint

Server Management System ^{1.0}

[Base URL: localhost:8080]
doc.json

A server management service API in Go using Gin framework.

Nguyen Minh Manh - Website
Send email to Nguyen Minh Manh
Apache 2.0

Authorize 

Server CRUD ^

POST	/api/server/	Create new server	▼	🔒
GET	/api/server/{id}	Get server by ID	▼	🔒
PUT	/api/server/{id}	Update server	▼	🔒
DELETE	/api/server/{id}	Delete server by ID	▼	🔒
GET	/api/servers	Export servers	▼	🔒
POST	/api/servers	Import servers	▼	🔒
GET	/api/servers/report	Report server information intentionally	▼	🔒

Login ^

POST	/login	Login Handler	▼	
------	--------	---------------	---	--

2. Sử dụng Unit Test

- Tạo ra Mock Database, sử dụng theo db chính mỗi khi test
- Tạo ra các mock get request để mô phỏng các get request
- Tạo ra các mock post request để mô phỏng các post request
- Tạo ra các mock put request để mô phỏng các put request
- Tạo ra các mock delete request để mô phỏng các delete request

Demo:

file test của endpoint create new server:

```

func TestCreateNewServer(t *testing.T) {
    // Load env variables
    err := godotenv.Load()
    if err != nil {
        godotenv.Load("../.env")
    }

    mock_dbName = os.Getenv("MOCK_DBNAME")
    dbName = os.Getenv("DBNAME")

    // Create mock database
    test.MockedDB(test.CREATE)

    // Replace current connection with the mock database connection
    serverDatabase.ReplaceConnection(mock_dbName)

    // Do our test

    var server = entity.Server{
        Name: "server1",
        Ipv4: "172.22.0.11",
        User: "root",
        Password: "helloworld",
        Status: "Up",
    }

    w := httptest.NewRecorder()
    ctx := test.GetTestGinContext(w)

    test.MockSimplePost(ctx, server)

    CreateNewServer(ctx)

    assert.EqualValues(t, http.StatusOK, w.Code)

    // Drop mock database
    defer test.MockedDB(test.DROP)

    // Replace mock database connection with the real one
    serverDatabase.ReplaceConnection(dbName)
}

```

1. Tạo mock database

2. Sử dụng mock post request để mô phỏng quá trình gọi api tạo server

3. Xóa mock database

2. Xác thực phân quyền bằng JWT

Người dùng sử dụng endpoint /login để lấy token, cần thêm các token này vào các request để có thể thực hiện thành công

Login			^
POST	/login	Login Handler	✓

3. Redis Cache

Sử dụng Redis Cache để optimize performance cho các nghiệp vụ view server, export server

```
redis:
  image: redis:7.0.9-alpine
  restart: always
  ports:
    - '6379:6379'
  command: redis-server --save 20 1 --loglevel warning
  environment:
    - ALLOW_EMPTY_PASSWORD=yes
  volumes:
    - cache:/data
```

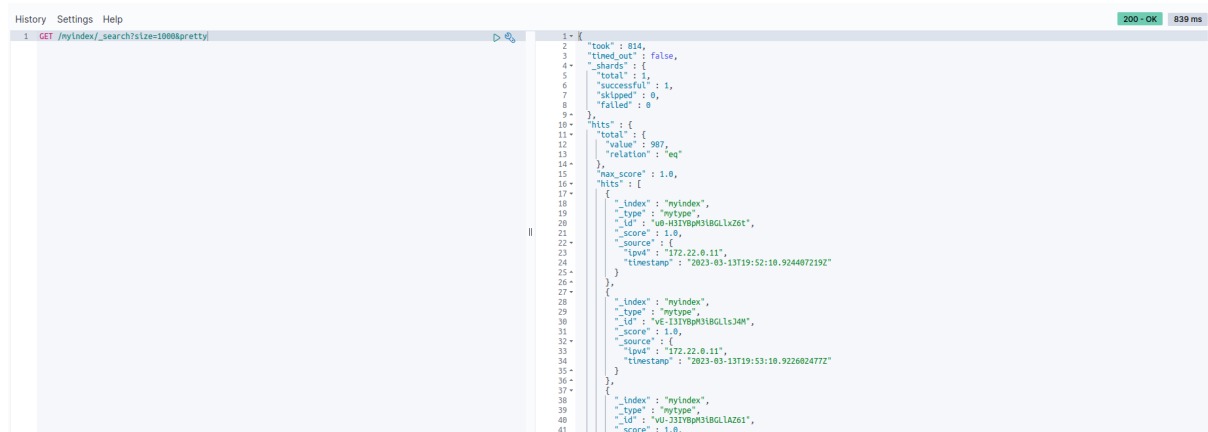
Thêm redis vào hệ thống

```
func (controller *serverController) ViewServer(id int) (entity.Server, error) {
    var server *entity.Server = controller.serverCache.Get(fmt.Sprintf(id))
    if server == nil {
        server, err := controller.serverService.ViewServer(id)
        controller.serverCache.Set(fmt.Sprintf(id), &server)
        return server, err
    }
    return *server, nil
}
```

Thực hiện kiểm tra dữ liệu cache trước mỗi request

4. Elasticsearch

Dùng để lưu trữ các phản hồi từ các server đến SMS server, dùng để tổng hợp thông tin báo cáo định kỳ



```
1 GET /myindex/_search?size=10000&pretty
2 {
3   "took": 814,
4   "timed_out": false,
5   "shards": {
6     "total": 1,
7     "successful": 1,
8     "skipped": 0,
9     "failed": 0
10  },
11  "hits": {
12    "total": {
13      "value": 987,
14      "relation": "eq"
15    },
16    "max_score": 1.0,
17    "hits": [
18      {
19        "_index": "myindex",
20        "_type": "mytype",
21        "_id": "u0-H3IV8pt91BQLxZ6t",
22        "_score": 1.0,
23        "_source": {
24          "ipV4": "172.22.0.11",
25          "timestamp": "2023-03-13T19:52:10.924407219Z"
26        }
27      },
28      {
29        "_index": "myindex",
30        "_type": "mytype",
31        "_id": "u0-H3IV8pt91BQLxZ6t",
32        "_score": 1.0,
33        "_source": {
34          "ipV4": "172.22.0.11",
35          "timestamp": "2023-03-13T19:53:10.922602477Z"
36        }
37      },
38      {
39        "_index": "myindex",
40        "_type": "mytype",
41        "_id": "u0-H3IV8pt91BQLxZ6t",
42        "_score": 1.0
```

5. Log và Logrotate

Ta tạo middle sử dụng thư viện zerolog


```

func StructuredLogger(logger *zerolog.Logger) gin.HandlerFunc {
    return func(ctx *gin.Context) {

        start := time.Now() // Start timer
        path := ctx.Request.URL.Path
        raw := ctx.Request.URL.RawQuery

        // Process request
        ctx.Next()

        // Fill the params
        param := gin.LogFormatterParams{}

        param.TimeStamp = time.Now() // Stop timer
        param.Latency = param.TimeStamp.Sub(start)
        if param.Latency > time.Minute {
            param.Latency = param.Latency.Truncate(time.Second)
        }

        param.ClientIP = ctx.ClientIP()
        param.Method = ctx.Request.Method
        param.StatusCode = ctx.Writer.Status()
        param.ErrorMessage = ctx.Errors.ByType(gin.ErrorTypePrivate).String()
        param.BodySize = ctx.Writer.Size()
        if raw != "" {
            path = path + "?" + raw
        }
        param.Path = path

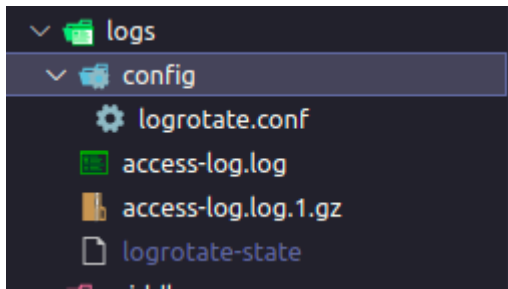
        // Log using the params
        var logEvent *zerolog.Event
        if ctx.Writer.Status() >= 500 {
            logEvent = logger.Error()
        } else {
            logEvent = logger.Info()
        }

        logEvent.Str("client_id", param.ClientIP).
            Str("method", param.Method).
            Int("status_code", param.StatusCode).
            Int("body_size", param.BodySize).
            Str("path", param.Path).
            Str("latency", param.Latency.String()).
            Msg(param.ErrorMessage)
    }
}

```

Và sử dụng service logservice của ubuntu để tự động tạo mới file log, lưu trữ và xóa các file log cũ sau một thời gian nhất định.

Các config được thiết lập trong file logrotate.conf



Nội dung của file log

```
access-log.log
{"level":"info","client_id":"172.22.0.10","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.10","latency":"1.200353ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.11","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.11","latency":"25.428776ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.12","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.12","latency":"30.948436ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.9","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.9","latency":"30.899453ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.8","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.8","latency":"855.57µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.12","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.12","latency":"1.007118ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.9","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.9","latency":"1.577173ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.10","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.10","latency":"10.215895ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.11","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.11","latency":"11.867024ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.8","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.8","latency":"996.987µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.10","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.10","latency":"956.369µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.12","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.12","latency":"890.587µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.11","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.11","latency":"1.000959ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.9","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.9","latency":"20.993212ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.8","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.8","latency":"573.667µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.10","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.10","latency":"529.409µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.11","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.11","latency":"445.615µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.12","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.12","latency":"15.525561ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.9","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.9","latency":"20.380795ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.8","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.8","latency":"1.001185ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.10","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.10","latency":"945.704µs","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.9","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.9","latency":"31.654416ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.11","method":"GET","status_code":200,"body_size":16,"path":"/checkExistence/172.22.0.11","latency":"40.618779ms","time":1645441600.000000000}
{"level":"info","client_id":"172.22.0.12","method":"GET","status_code":200,"body_size":17,"path":"/checkExistence/172.22.0.12","latency":"40.887853ms","time":1645441600.000000000}
```

VI. Cách cài đặt:

1. Hệ thống có sẵn docker và docker compose
2. Download mã nguồn
3. Chạy câu lệnh docker compose up
4. Truy cập địa chỉ localhost:8080 để truy cập server