**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN THỰC HÀNH SỐ 1**

**XÂY DỰNG MÁY CHỦ WEB SỬ DỤNG UNIX SOCKET**

**Học phần: Lập trình Mạng**

**Lớp: CQ2019/4**

**Họ và tên các thành viên:**

1. **Lê Văn – 1712897**
2. **Phạm Ngọc Minh – 18120059**
3. **Trần Trung Hậu – 18120360**
4. **Bùi Lê Tuấn Anh – 19120163**
5. **Phạm Anh Khoa – 19120257**

***Thành phố Hồ Chí Minh, tháng 11 năm 2022***

Nội dung

[1. GIỚI THIỆU CHUNG 1](#_Toc120901947)

[2. PHÂN CÔNG TRIỂN KHAI 3](#_Toc120901948)

[3. CÁC BƯỚC TRIỂN KHAI 5](#_Toc120901949)

[a. Phát triển mã nguồn trang web 5](#_Toc120901950)

[b. Phát triển mã nguồn máy chủ 6](#_Toc120901951)

[1. Khai báo và hàm main 7](#_Toc120901952)

[2. Hàm kiểm tra địa chỉ IP 12](#_Toc120901953)

[3. Hàm xử lý định dạng tập tin 13](#_Toc120901954)

[4. Các hàm xử lý lệnh GET 14](#_Toc120901955)

[5. Các hàm xử lý lệnh POST 15](#_Toc120901956)

[c. Phát triển Makefile và chùm tập tin Docker 17](#_Toc120901957)

[1. Makefile 17](#_Toc120901958)

[2. Dockerfile 18](#_Toc120901959)

[3. Docker Compose 18](#_Toc120901960)

[4. NHẬN XÉT VÀ ĐÁNH GIÁ KẾT QUẢ 21](#_Toc120901961)

[a. Kết quả chương trình 21](#_Toc120901962)

[b. Đánh giá chi tiết 23](#_Toc120901963)

[c. Hướng phát triển tiếp theo 23](#_Toc120901964)

[TÀI LIỆU THAM KHẢO 25](#_Toc120901965)

# GIỚI THIỆU CHUNG

Đối với Mạng máy tính, Socket luôn là một trong những thành phần không thể thiếu.

# PHÂN CÔNG TRIỂN KHAI

Dưới đây là bảng phân công và đánh giá công việc cho từng thành viên.

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Công việc** | **Mức độ** |
| **1** | Lê Văn | * Hỗ trợ đảm trách phần báo cáo. | 0% |
| **2** | Trần Trung Hậu | * **Phụ trách kết nối API** * Đảm trách phần lập trình chính cho mã nguồn của máy chủ (Socket) | 50% |
| **3** | Phạm Ngọc Minh | * Đảm trách chính phần slides trình chiếu cho nhóm. * Hỗ trợ nhóm lập trình trong các vấn đề khó liên quan | 50% |
| **4** | Bùi Lê Tuấn Anh | * **Nhóm trưởng và người thuyết trình** * Đảm trách mã nguồn trang web cho máy chủ. * Đảm trách chính báo cáo. * Đảm trách viết Makefile, Dockerfile, Docker Compose và thử nghiệm trên máy chủ điện toán đám mây. | 100% |
| **5** | Phạm Anh Khoa | * **Nhóm phó phụ trách lập trình** * Cùng đảm trách phần lập trình chính cho mã nguồn của máy chủ (Socket) | 100% |
|  | **Tổng cộng** |  | **60%** |

Dưới đây là phần tự đánh giá điểm số của các thành viên:

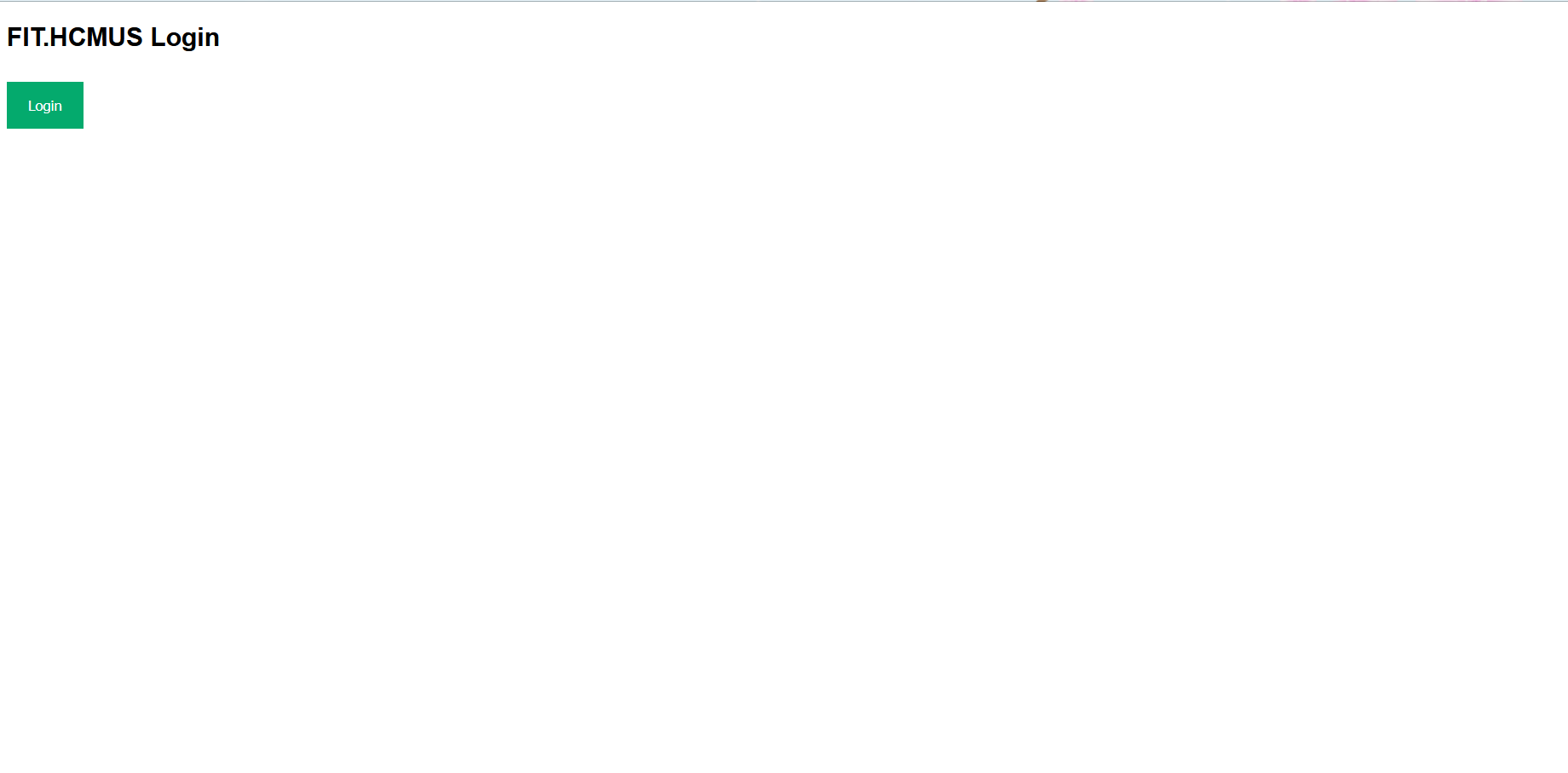
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **Mức độ hoàn thành** | **Số điểm (trên 10)** |
| **1** | Lê Văn | 0% | 7 |
| **2** | Trần Trung Hậu | 30% | 8 |
| **3** | Phạm Ngọc Minh | 50% | 8 |
| **4** | Bùi Lê Tuấn Anh | 70% | 9 |
| **5** | Phạm Anh Khoa | 100% | 9 |

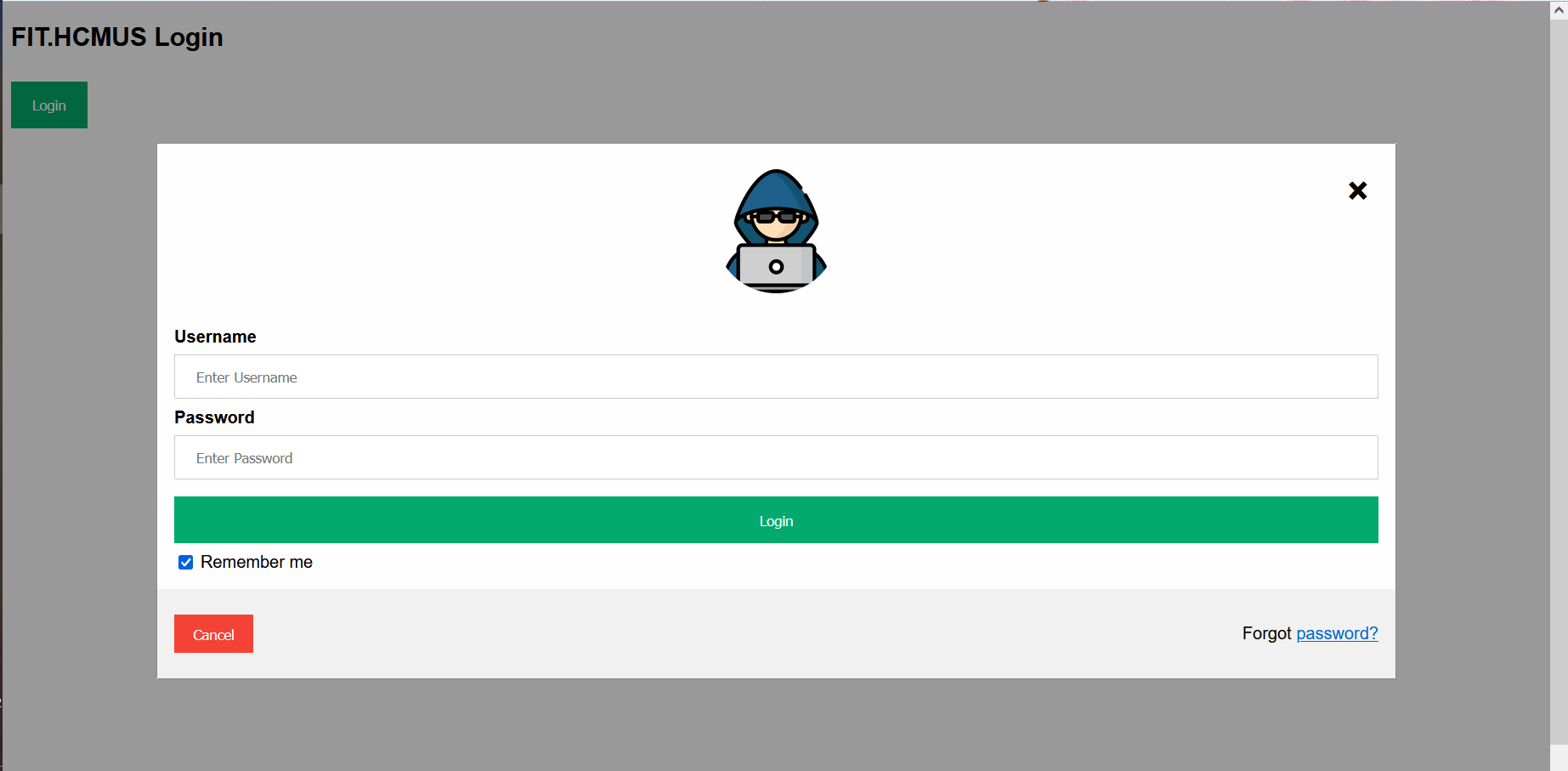
# CÁC BƯỚC TRIỂN KHAI

## Phát triển mã nguồn trang web

Mã nguồn của trang web đã được cung cấp sẵn cho báo cáo này, trong đó bao gồm các trang như sau:

* **Trang index.html**: Đây là trang đăng nhập chính thức cho hệ thống.
  + Trang này ngoài giao diện được tải từ tập tin **style.css** ra còn có thêm đoạn mã nguồn JavaScript dùng để xử lý khi người dùng nhấn vào nút Login trên màn hình giao diện.





* + Khi người dùng bấm nút **Login**, hệ thống chuyển sang trang đăng nhập với ảnh đại diện được tải ngẫu nhiên cùng 1 biểu mẫu đăng nhập để vào tiếp các trang khác. Biểu mẫu này chấp nhận tên đăng nhập và mật khẩu cố định.
* **Trang images.html**: Trang cung cấp hình ảnh. Trang này tải tập tin **utils.css** để tạo ra giao diện phân ô cho các hình ảnh.



* + Với trang này, sau khi người dùng **đăng nhập thành công** thì sẽ chuyển tới để trình chiếu các hình ảnh được tải lên từ hệ thống. Trong trường hợp nếu **không đăng nhập thành công**, hệ thống báo lỗi và tự động trả về trang **index.html** theo đung quy định.

## Phát triển mã nguồn máy chủ

Toàn bộ mã nguồn của máy chủ đều sử dụng **socket**, viết bằng ngôn ngữ lập trình **C** và được tổ chức thành các tập tin và hàm như sau:

### Khai báo và hàm main

Ảnh có chứa văn bản, tấm

Mô tả được tạo tự động

Đây là tập tin chứa toàn bộ khai báo tất cả các thư viện sẽ sử dụng trong chương trình, đồng thời cũng chứa những định nghĩa về giá trị các biến bao gồm:

* Cổng sử dụng: **8080 (không dùng cổng 80 trực tiếp, nhưng sẽ chuyển tiếp về cổng 80)**
* Kích thước bộ nhớ đệm: **1024 Bytes**
* Chuỗi ký tự phân tách header và body của trang web: **“\r\n\r\n”**

Ảnh có chứa văn bản, màn hình

Mô tả được tạo tự động

Đoạn hàm này xác định các tiêu chí sau:

* *Header của các lệnh GET.*
* *Header của các lệnh POST.*
* *Nội dung bên trong của Header và Bộ nhớ đệm.*
* *Con trỏ đọc nội dung tập tin.*
* *Cờ xác định việc đăng nhập hay chưa.*
* *Biến tùy chọn.*
* *Biến xác định kích thước tập tin.*

Ảnh có chứa văn bản

Mô tả được tạo tự động

Đoạn hàm này khởi tạo socket và kiểm tra việc khởi tạo socket có diễn ra thành công hay không, sau đó gửi thông báo cho người dùng.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Đoạn hàm này khởi tạo địa chỉ cũng như gán địa chỉ và cổng vận hành vào các máy chủ để hiển thị ra bên ngoài cho người dùng.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Đoạn hàm này khởi tạo địa chỉ máy khách, cũng như gắn socket vào địa chỉ để thực hiện việc lắng nghe kết nối từ bên ngoài vào bên trong các máy chủ.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Đoạn hàm này chấp nhận các kết nối từ máy khách vào máy chủ, đồng thời ghi nhận lại địa chỉ các máy khách kết nối vào máy chủ. Báo lỗi nếu không nhận được thông tin kết nối hay không chấp nhận kết nối.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Đoạn hàm này cho phép việc đọc nội dung từ socket và lấy thông tin quan trọng, bao gồm phương thức (GET, POST) và địa chỉ sử dụng phương thức.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản, màn hình, ảnh chụp màn hình

Mô tả được tạo tự động

Đoạn hàm này có tác dụng kiểm tra điều kiện sau:

* Nếu phương thức truy cập là GET: ***sao chép toàn bộ header của GET được định nghĩa trong main vào header chính*** và kiểm tra đường dẫn đầu vào:
  + Nếu đường dẫn đầu vào là **“/”**, tiến hành đóng gói và xử lý gói tin GET cho trang **index.html**
  + Nếu đường dẫn đầu vào là “/images.html” và **Chưa đăng nhập**: Điều hướng trở lại trang **index.html**
  + Xử lý đồng thời các tập tin CSS, JPG, PNG trong web.
* Nếu phương thức truy cập là POST: ***sao chép toàn bộ header của POST được định nghĩa trong main vào header chính*** và kiểm tra việc đăng nhập:
  + **Đã đăng nhập**: Tiến hành đóng gói và xử lý tiếp POST cho trang.
  + **Chưa đăng nhập**: Tiến hành quay đầu và xử lý đường dẫn “/”.

### Hàm kiểm tra địa chỉ IP

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hàm này có công dụng chỉ ra được địa chỉ của máy chủ, từ đó xác định được địa chỉ để các máy khách có thể truy cập được.

### Hàm xử lý định dạng tập tin

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hàm này có công dụng kiểm tra định dạng của các tập tin và phân định cho việc đọc nội dung các tập tin (HTML, JPG, CSS, PNG, …)

### Các hàm xử lý lệnh GET

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hàm này sẽ thực hiện các yêu cầu sau:

* **Bước 1**: *Đọc nội dung tập tin cần gửi cho máy khách vào một con trỏ được khởi tạo trước đó*
* **Bước 2**: *Gắn thêm thuộc tính Content-Length ứng với kích thước của phần nội dung gửi vào header.*
* **Bước 3**: *Gắn thêm thuộc tính mimeType ứng với định dạng tập tin cần gửi vào header.*
* **Bước 4**: *Gắn thêm chuỗi ký tự phân tách “\r\n\r\n” vào cuối phần header.*
* **Bước 5**: *Gọi hàm gửi lệnh GET.*

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hàm này thực hiện các bước sau:

* *Đọc header vào file descriptor*
* *Gán con trỏ nội dung vào con trỏ dữ liệu*
* *Kiểm tra kích thước nội dung: Nếu nhỏ hơn bộ đệm thì đọc toàn bộ, nếu ít hơn thì đọc đúng kích thước bộ đệm, sau đó đọc tuần tự.*
* *Sau khi đọc xong: dọn dẹp bộ nhớ, đóng file descriptor.*

### Các hàm xử lý lệnh POST

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hàm này có công dụng kiểm tra xem người dùng đã đăng nhập vào hệ thống hay chưa, nếu đã đăng nhập thành công thì bật cờ hiệu để báo hiệu cho hệ thống.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hàm này thực hiện các nhiệm vụ sau:

* **Bước 1**: *Xác định thư mục cần gửi cho máy khách.*
* **Bước 2**: *Xác định thuộc tính Content-Type ứng với định dạng tập tin cần gửi trước khi gán thuộc tính cho header.*
* **Bước 3**: *Gắn thêm chuỗi ký tự phân tách “\r\n\r\n” vào cuối phần header.*
* **Bước 4**: *Gọi hàm gửi lệnh POST đến hệ thống.*

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hàm này thực hiện các bước sau:

* *Đọc header vào file descriptor*
* *Sau khi đọc xong: dọn dẹp bộ nhớ, đóng file descriptor.*

Bởi vì nội dung của phương thức POST không chứa dữ liệu body bên trong, do đó chỉ cần đọc header tương ứng với phương thức là đủ, toàn bộ dữ liệu còn lại đều do máy chủ xử lý mà không cần làm lộ ra dữ liệu.

## Phát triển Makefile và chùm tập tin Docker

### Makefile

Với Makefile, các tập tin mã nguồn sẽ được biên dịch và tạo ra chỉ một tập tin duy nhất là **tập tin thực thi chương trình**. Dựa trên kiến trúc và mã nguồn đã được xây dựng, nội dung Makefile được cụ thể hóa như sau:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Các câu lệnh chạy đều đã được chuẩn hóa như hình trên, bao gồm:

* **All***: Chạy biên dịch tất cả tập tin đối tượng thành chương trình hoàn chỉnh.*
* **Clean***: Dọn dẹp sau biên dịch chương trình.*
* *Các câu lệnh makedir dùng để tạo thư mục chứa các tập tin.*
* *Các câu lệnh thực hiện biên dịch ra các tập tin chứa dạng đối tượng (.o) để biên dịch thành chương trình hoàn chỉnh.*
* *Sử dụng lệnh* ***make*** *(hoặc* ***make all****) để thực hiện biên dịch trước khi chuyển sau phần tiếp theo.*

### Dockerfile

Docker là một công cụ mã nguồn mở giúp triển khai mã nguồn các ứng dụng Linux và Windows vào trong môi trường container ảo hóa. Đối với báo cáo này, để có thể kiểm thử khả năng vận hành của máy chủ thì sau khi biên dịch xong chương trình, **tập tin thực thi và mã nguồn trang web** được sao chép vào **Docker container** để có thể ảo hóa hoạt động của một máy chủ websocket. Cụ thể như sau:

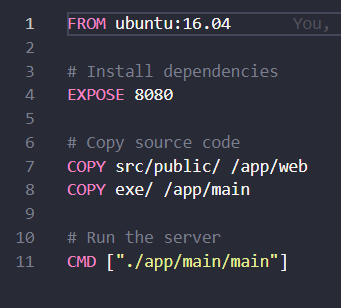


Image để sử dụng cho các Docker container cũng đã được đính kèm lên kho lưu trữ mã nguồn Github của nhóm, với tính năng **Github Container Registry**. Lúc này, toàn bộ các **Docker image** đều có thể lưu lại **một cách miễn phí (đi kèm với giới hạn dung lượng)** và có thể tải xuống để sử dụng một cách dễ dàng và ít tốn kém nhất cho các mục tiêu **mở rộng quy mô** cho hệ thống kiểm thử.

### Docker Compose

Nếu căn cứ vào mô hình của bài toán ban đầu, để triển khai một hệ thống đa máy chủ thì không thể chỉ chạy **1 Docker Container**, bởi số lượng lúc này cần phải là **3**. Do đó, cần có thêm một công cụ để giả lập quá trình nhân bản số lượng máy chủ theo ý muốn của chúng ta, và công cụ tương thích với Docker nhất có thể sử dụng chính là **Docker Compose**.

Toàn bộ nội dung của tập tin Docker Compose như sau:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Tập tin này trước hết kích hoạt và tạo lập một mạng có địa chỉ **172.10.0.0/16**, sau đó tiến hành lấy image Docker về, mở kết nối cổng **8080** và nhân bản image này ra **3 container (ứng với 3 máy chủ)** sử dụng mạng đã được tạo lập trước đó để vận hành hệ thống máy chủ.

Tại thời điểm viết báo cáo này, để truy cập vào hệ thống máy chủ này thì sử dụng 3 cổng khác nhau với cùng địa chỉ IP để chuyển tiếp đến địa chỉ của máy chủ. Mô hình này tương tự như hệ thống cân bằng tải được sử dụng trong thực tế hay gần gũi hơn là mô hình **ingress trong Kubernetes với tính năng NodePort**.

# NHẬN XÉT VÀ ĐÁNH GIÁ KẾT QUẢ

## Kết quả chương trình

Dưới đây là kết quả chương trình, sau khi được triển khai trên hệ thống đa máy chủ: ***Sử dụng cùng 1 địa chỉ duy nhất, truy cập vào ba máy chủ được chuyển tiếp cổng sang 3 cổng khác nhau để cho ra kết quả như hình dưới***.

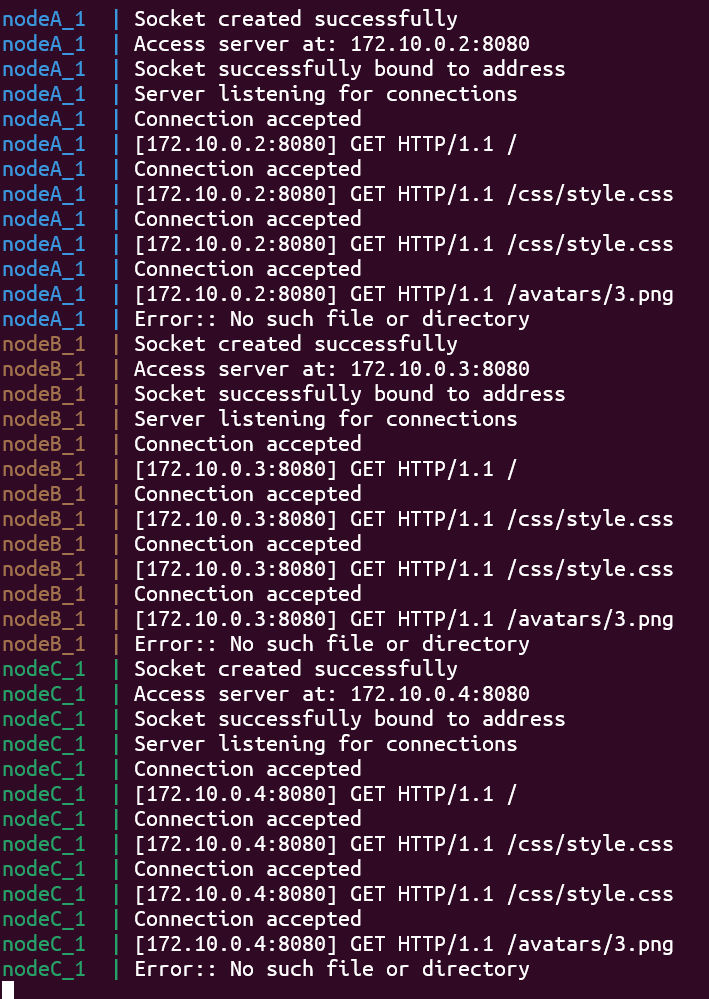
Ảnh có chứa văn bản, tấm

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Thử trên hệ thống máy chủ chính thức:



## Đánh giá chi tiết

* Dựa trên log này, nhóm có một số đánh giá như sau:
  + ***Chương trình có khả năng xử lý được các trang web khác nhau, tại các máy chủ khác nhau, với kiểu Connection là Keep-Alive.***
  + ***Việc lắng nghe kết nối đối với socket diễn ra một cách chính xác mà không gặp phải vấn đề nghiêm trọng.***
  + ***Các hình ảnh và tập tin CSS cũng như các tập lệnh JavaScript chứa trong tập tin HTML đều được xử lý một cách chuẩn xác mà không gặp phải sai số.***
* Tuy nhiên, vẫn còn đó những vấn đề phát sinh như:
  + ***Chưa xử lý được việc giữ token.*** Mã nguồn hiện tại chưa cho phép gửi token cho người dùng sau khi đăng nhập, mà nguyên nhân một phần cũng do chính mã nguồn trang web chính thức.
  + ***Chưa xử lý cho nhiều tài khoản đăng nhập.*** Vấn đề này có thể dẫn đến việc nhiều máy khách cùng đăng nhập vào hệ thống với cùng một tài khoản dẫn đến việc có nhiều người có cùng quyền truy cập vào hệ thống, gây lỗ hổng về bảo mật.

## Hướng phát triển tiếp theo

Dựa trên đánh giá về những vấn đề phát sinh trong quá trình thực hiện báo cáo này, nhóm đề xuất một số hướng phát triển sau:

* ***Triển khai tính năng ghi nhớ đăng nhập (Remember me)*** *để phục vụ cho việc hạn chế số lần đăng nhập vào hệ thống, tiết kiệm thời gian****.***
* ***Triển khai việc xử lý danh sách người dùng*** *để giúp cho nhiều người dùng có thể sử dụng dịch vụ thay vì chỉ có một người dùng như hiện tại****.***
* ***Triển khai việc xử lý gửi token cho từng người dùng đăng nhập*** *để giúp cho việc truy nhập được diễn ra bảo mật và an toàn hơn****.***

# TÀI LIỆU THAM KHẢO

* <https://gist.github.com/natcl/3d881d00a56c8a961e6dab8ba51a5a37>
* <https://docs.docker.com/compose/gettingstarted/>
* <https://docs.docker.com/compose/compose-file/compose-file-v3/>
* <https://dev.to/fmtweisszwerg/cc-how-to-get-all-interface-addresses-on-the-local-device-3pki>
* <https://emalsha.wordpress.com/2016/11/24/how-create-http-server-using-python-socket-part-ii/>
* <https://www.appsloveworld.com/c/100/253/c-sending-image-through-socket>
* <https://bruinsslot.jp/post/simple-http-webserver-in-c/>
* <https://stackoverflow.com/questions/8257714/how-to-convert-an-int-to-string-in-c>
* <https://www.geeksforgeeks.org/socket-programming-cc/>
* <https://stackoverflow.com/questions/24194961/how-do-i-use-setsockoptso-reuseaddr>
* <https://github.com/AtmegaBuzz/weebserver>
* <https://github.com/TheHellhand/Socket_WebServer>