

**ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

----

**BÁO CÁO CUỐI KỲ
QUẢN TRỊ MẠNG VÀ HỆ THỐNG**

Đề tài: Tìm hiểu và triển khai ứng dụng trên Docker



Giảng viên hướng dẫn	:	Đỗ Quang Hiển
Lớp	:	NT132.O11.ANTT
Sinh viên thực hiện	:	Nguyễn Thị Hồng Lam 20521518 Lê Đào Khánh Ngọc 20520653 Mai Ngọc Phương Trinh 20520823 Trần Lê Minh Ngọc 21521195 Huỳnh Minh Khuê 21522240

TP.HCM năm 2023

**ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

----

**BÁO CÁO CUỐI KỲ
QUẢN TRỊ MẠNG VÀ HIỆT THÔNG**

Đề tài: Tìm hiểu và triển khai ứng dụng trên Docker



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn	:	Đỗ Quang Hiển
Lớp	:	NT132.O11.ANTT
Sinh viên thực hiện	:	Nguyễn Thị Hồng Lam 20521518
		Lê Đào Khánh Ngọc 20520653
		Mai Ngọc Phương Trinh 20520823
		Trần Lê Minh Ngọc 21521195
		Huỳnh Minh Khuê 21522240

TP.HCM năm 2023

Lời cảm ơn

Chúng em xin chân thành cảm ơn Khoa Mạng máy tính và Truyền thông, Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em học tập và thực hiện đề tài báo cáo cuối kì này này.

Chúng em xin bày tỏ lòng biết ơn sâu sắc tới thầy Đỗ Quang Hiển đã tận tình hướng dẫn chỉ bảo chúng em trong quá trình thực hiện đề tài.

Mặc dù đã cố gắng hoàn thành bài báo cáo trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo của quý thầy cô và các bạn.

Chúng em xin chân thành cảm ơn quý thầy cô và các bạn!

Sinh viên

Nguyễn Thị Hồng Lam

Lê Đào Khánh Ngọc

Mai Ngọc Phương Trinh

Trần Lê Minh Ngọc

Huỳnh Minh Khuê

Mục lục

I – MỞ ĐẦU.....	1
II – CƠ SỞ LÝ THUYẾT.....	1
1. Docker	1
1.1. Docker là gì	1
1.2. Các thành phần của Docker	2
1.3. Kiến trúc Docker.....	2
1.4. Lý do nên sử dụng Docker.....	4
1.5. Một số câu lệnh trong Docker	4
1.6. Một số câu hỏi phổ biến về Docker	5
2. Docker Compose.....	5
2.1. Docker Compose là gì	5
2.2. Một số câu lệnh trong Docker Compose	6
III – TRIỂN KHAI ỨNG DỤNG.....	6
1. Ứng dụng trò chơi pygame.....	6
1.1. Revenge of the Rat	6
1.2. Flappybird.....	15
2. Ứng dụng MySQL	16
2.1. Giới thiệu về ứng dụng	16
2.2. Sử dụng Docker cho ứng dụng	17
2.3. Hướng dẫn cài đặt.....	19
3. Ứng dụng Blog cá nhân	21

3.1.	Giới thiệu về ứng dụng	21
3.3.	Sử dụng Docker cho ứng dụng	21
3.4.	Hướng dẫn cài đặt.....	25
3.5.	Công việc tương lai	30
4.	Ứng dụng ghi chú Todolist.....	30
4.1.	Giới thiệu về ứng dụng	30
4.2.	Sử dụng Docker cho ứng dụng	31
4.3.	Hướng dẫn cài đặt.....	32
IV – Hướng dẫn sửa lỗi		35
1.	Ứng dụng trò chơi “Revenge of the Rat”	35
1.1.	Lỗi không thể sử dụng giao diện và âm thanh của máy chủ	35
1.2.	Lỗi không thể chạy file main.py	36
2.	Ứng dụng Blog cá nhân	37
Phân công công việc.....		39
Tài liệu tham khảo.....		39

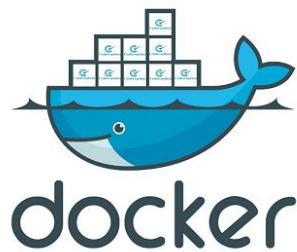
I – MỞ ĐẦU

Trong thời đại công nghệ phát triển vượt bậc, việc triển khai ứng dụng trở nên ngày càng phức tạp do sự đa dạng về môi trường hệ thống và yêu cầu ngày càng cao về linh hoạt và tính di động. Do đó, Docker, với sức mạnh của mình, đã nỗi lên như một giải pháp hiệu quả giúp đơn giản hóa quá trình triển khai và quản lý ứng dụng. Mục đích chính của việc sử dụng Docker không chỉ là giảm bớt sự phức tạp trong quá trình triển khai mà còn là tăng cường khả năng di động, khả năng mở rộng và quản lý tài nguyên hiệu quả. Bằng cách tận dụng công nghệ container, Docker không chỉ mang lại sự nhất quán giữa môi trường phát triển và môi trường sản xuất mà còn giúp tối ưu hóa tài nguyên hệ thống, giảm thiểu xung đột và tăng cường tính bảo mật. Triển khai ứng dụng trên Docker không chỉ là một xu hướng mà còn là một chiến lược quan trọng đối với các tổ chức và doanh nghiệp đang tìm kiếm sự linh hoạt và hiệu quả trong quản lý hạ tầng ứng dụng của mình. Điều này đã làm cho Docker trở thành một công cụ không thể thiếu trong bộ công cụ của các nhà phát triển và quản trị hệ thống, đồng thời mở ra những khả năng mới trong việc xây dựng và duy trì các ứng dụng hiện đại.

II – CƠ SỞ LÝ THUYẾT

1. Docker

1.1. Docker là gì



Docker là một nền tảng ảo hóa cấp cao sử dụng công nghệ container để đóng gói và triển khai ứng dụng. Mục tiêu chính của Docker là tạo ra môi trường đóng gói mà ở đó ứng

dụng và tất cả các phụ thuộc của nó (thư viện, biến môi trường, cấu hình, v.v.) được đóng gói lại thành một container duy nhất. Container này sau đó có thể chạy trên bất kỳ hệ thống nào đã cài đặt Docker mà không cần phải lo lắng về sự khác biệt trong môi trường hệ thống.

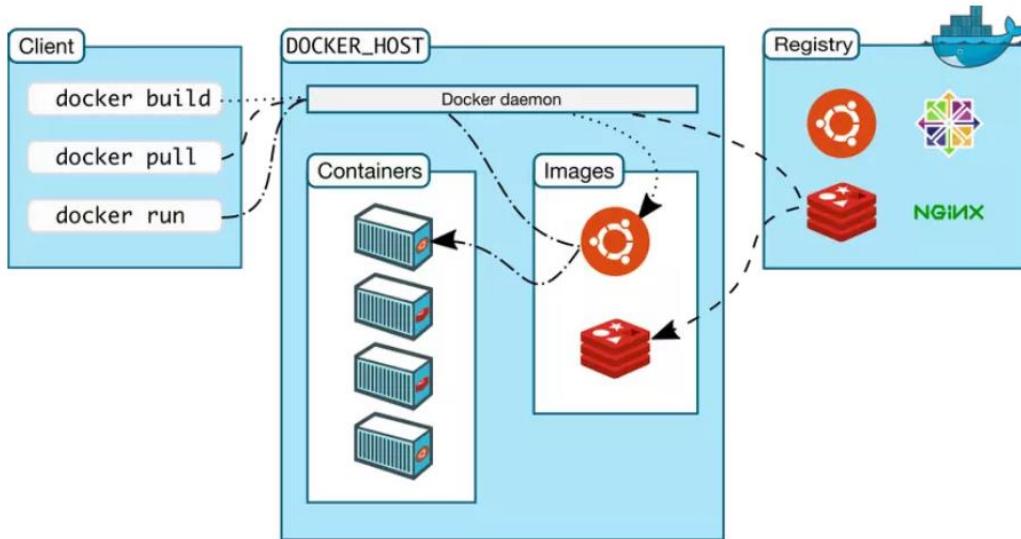
1.2. Các thành phần của Docker

Một số thành phần chính của Docker:

- Docker Engine: là thành phần chính của Docker, như một công cụ để đóng gói ứng dụng.
- Docker Hub: là một “github for docker images”. Trên DockerHub có hàng ngàn public images được tạo bởi cộng đồng cho phép người dùng dễ dàng tìm thấy những image mà mình cần.
- Images: là một khuôn mẫu để tạo một container. Người dùng có thể tự build một image riêng cho mình hoặc sử dụng những image được chia sẻ từ cộng đồng Docker Hub. Một image sẽ được build dựa trên những chỉ dẫn của Dockerfile.
- Container: là một instance của một image. Người dùng có thể create, start, stop, move or delete container dựa trên Docker API hoặc Docker CLI.
- Docker Client: là một công cụ giúp người dùng giao tiếp với Docker host.
- Docker Daemon: lắng nghe các yêu cầu từ Docker Client để quản lý các đối tượng như Container, Image, Network và Volumes thông qua REST API. Các Docker Daemon cũng giao tiếp với nhau để quản lý các Docker Service.
- Dockerfile: là một tập tin bao gồm các chỉ dẫn để build một image .
- Volumes: là phần dữ liệu được tạo ra khi container được khởi tạo.

1.3. Kiến trúc Docker

Docker sử dụng kiến trúc client-server.



Hình 1: Kiến trúc docker

Cả Docker client và Docker daemon có thể chạy trên cùng 1 máy, hoặc có thể kết nối theo kiểu Docker client điều khiển các docker daemon như hình trên. Docker client và daemon giao tiếp với nhau thông qua socket hoặc RESTful API.

Docker Daemon (dockerd):

- Docker Daemon là một tiến trình chạy ngầm trên hệ điều hành chủ (host OS).
- Nhiệm vụ của Docker Daemon là quản lý các container, hình ảnh (images), volumes, và các thành phần khác của Docker.
- Thực hiện các tác vụ build, run và distributing các Docker container.
- Được điều khiển thông qua Docker API.
- Docker daemon chạy trên các máy host. Người dùng sẽ không tương tác trực tiếp với các daemon, mà thông qua Docker Client.

Docker Client:

- Docker Client là một công cụ dòng lệnh hoặc giao diện người dùng đồ họa (Docker CLI hoặc Docker Dashboard).
- Khi người dùng nhập các lệnh Docker, Docker Client gửi yêu cầu đến Docker Daemon để thực hiện các hành động tương ứng.

Docker Registries:

- Docker Registries là nơi lưu trữ và chia sẻ Docker Images.
- Docker Hub là một public registry phổ biến, nhưng bạn cũng có thể sử dụng private registries hoặc tự triển khai một registry của mình.

1.4. Lý do nên sử dụng Docker

Việc triển khai Docker đem lại cho người dùng rất nhiều lợi ích, hãy cùng điểm qua một số ưu điểm nổi bật của Docker:

- Vận chuyển phần mềm nhiều hơn và nhanh hơn: Người dùng sử dụng Docker vận chuyển phần mềm nhanh hơn trung bình 7 lần so với người dùng không sử dụng Docker. Docker đem đến khả năng vận chuyển dịch vụ được tách riêng với tần suất mong muốn.
- Tiêu chuẩn hóa quá trình vận hành: Ứng dụng được đóng gói vào container nhỏ sẽ khiến cho việc triển khai, xác định vấn đề và đảo ngược để khắc phục trở nên dễ dàng.
- Di chuyển tron tru: Ứng dụng trên nền tảng Docker có thể được di chuyển tron tru từ các máy phát triển cục bộ đến đơn vị triển khai. Với Docker, người dùng sẽ được nhận một đối tượng duy nhất có khả năng chạy ổn định ở bất kỳ đâu. Cú pháp đơn giản và không phức tạp của Docker sẽ cho người dùng quyền kiểm soát hoàn toàn.
- Tiết kiệm tiền bạc: Container Docker giúp cho việc chạy nhiều mã hơn trên cùng máy chủ trở nên dễ dàng hơn, cải thiện khả năng tận dụng và tiết kiệm tiền bạc cho người dùng.

1.5. Một số câu lệnh trong Docker

Một số câu lệnh thông dụng trong Docker:

docker ps: liệt kê danh sách containers đang chạy.

docker ps -a: liệt kê danh sách tất cả các container kể cả đã stop.

docker pull: tải một Docker image từ Docker Hub registry.

docker build: được dùng để tạo ra một image dựa trên một file Dockerfile.

docker images hoặc docker image ls: hiển thị danh sách image ở máy người dùng.

docker run: chạy một container từ một image.

docker logs: hiển thị logs của một container mà người dùng chỉ ra.

docker volume ls: hiển thị danh sách volumes.

docker network ls: liệt kê tất cả các network có sẵn.

docker network: connect vào một network.
docker rm: loại bỏ một hoặc nhiều container.
docker rmi: xóa bỏ một hay nhiều image.
docker stop: dừng một hay nhiều container.
docker start: bắt đầu một container đã được dừng với trạng thái giữ nguyên.
docker update --restart=no: cập nhật một cài đặt container.
docker cp: sao chép các file từ một container đang chạy ra ngoài host.

1.6. Một số câu hỏi phổ biến về Docker

Câu hỏi 1: Container Windows có chạy được trên hệ điều hành Linux không?

Trả lời:

Container Windows không thể chạy được trên hệ điều hành Linux. Containers tận dụng kernel và tài nguyên của hệ điều hành máy chủ, điều này có nghĩa là hệ điều hành của container phải khớp với hệ điều hành của máy chủ. Containers Windows được thiết kế để chạy trên máy chủ Windows, và containers Linux được thiết kế để chạy trên máy chủ Linux.

Câu hỏi 2: Có thể chạy container Linux trên hệ điều hành Windows không?

Trả lời:

Container Linux có thể chạy được trên hệ điều hành Windows dựa trên một tính năng là Windows Subsystem for Linux. Windows Subsystem for Linux (WSL) là một tính năng của hệ điều hành Windows, được giới thiệu bởi Microsoft, cho phép bạn chạy một hệ điều hành Linux trực tiếp trên Windows mà không cần máy ảo hoặc khởi động lại máy tính. WSL giúp cầu nối sự khác biệt giữa hai hệ điều hành này và mang lại trải nghiệm tích hợp giữa môi trường phát triển Linux và hệ điều hành Windows.

2. Docker Compose

2.1. Docker Compose là gì

Docker Compose là một công cụ điều khiển Docker được sử dụng để đơn giản hóa quá trình quản lý và triển khai ứng dụng đa-container. Nó cho phép người dùng định nghĩa, cấu hình và khởi chạy đồng thời nhiều container Docker từ một tập tin cấu hình duy nhất, thường được gọi là "docker-compose.yml". Bằng cách này, Docker Compose giúp đơn giản hóa và tự động hóa quá trình triển khai các môi trường phức tạp.

Không giống như Dockerfile (build các image). Docker compose dùng để build và run các container.

2.2. Một số câu lệnh trong Docker Compose

Dưới đây là một số câu lệnh thông dụng trong Docker Compose:

docker-compose up: Khởi động các container

docker-compose down: Dừng và xóa các container

docker-compose ps: Hiển thị trạng thái của các container

docker-compose build: Tạo image từ Dockerfile trong mỗi dịch vụ

docker-compose restart: Khởi động lại các container

docker-compose stop: Dừng các container

docker-compose rm: Xóa các container không sử dụng

docker-compose logs: Hiển thị các logs của các container

docker-compose config: Hiển thị các cấu hình của Docker Compose

docker-compose exec: Thực thi một lệnh trên một container

docker-compose port: Hiển thị các port của các container

docker-compose top: Hiển thị các process đang chạy trong các container

III – TRIỂN KHAI ỨNG DỤNG

1. Ứng dụng trò chơi pygame

1.1. Revenge of the Rat

1.1.1. Giới thiệu về ứng dụng

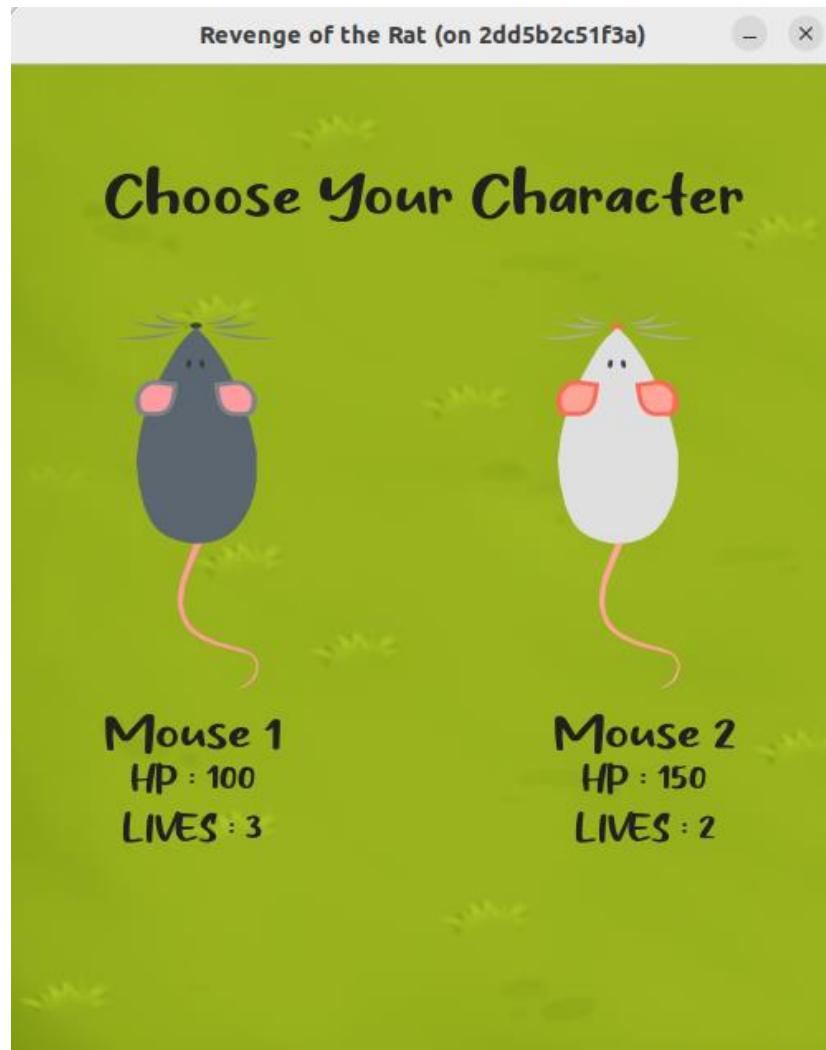
Mục đích của ứng dụng này là sử dụng Docker để làm môi trường tạo ra một ứng dụng trò chơi đơn giản có tên là “Revenge of the Rat” sử dụng thư viện pygame của python. “Revenge of the Rat” là một dạng trò chơi cho phép người chơi sử dụng bàn phím và chuột điều khiển nhân vật “chuột” trong trò chơi tiêu diệt các nhân vật “mèo”. Trò chơi có sử dụng giao diện đồ họa và âm thanh.

(tham khảo <https://github.com/riecho14/Docker-Dendam-Si-Tikus.git>).

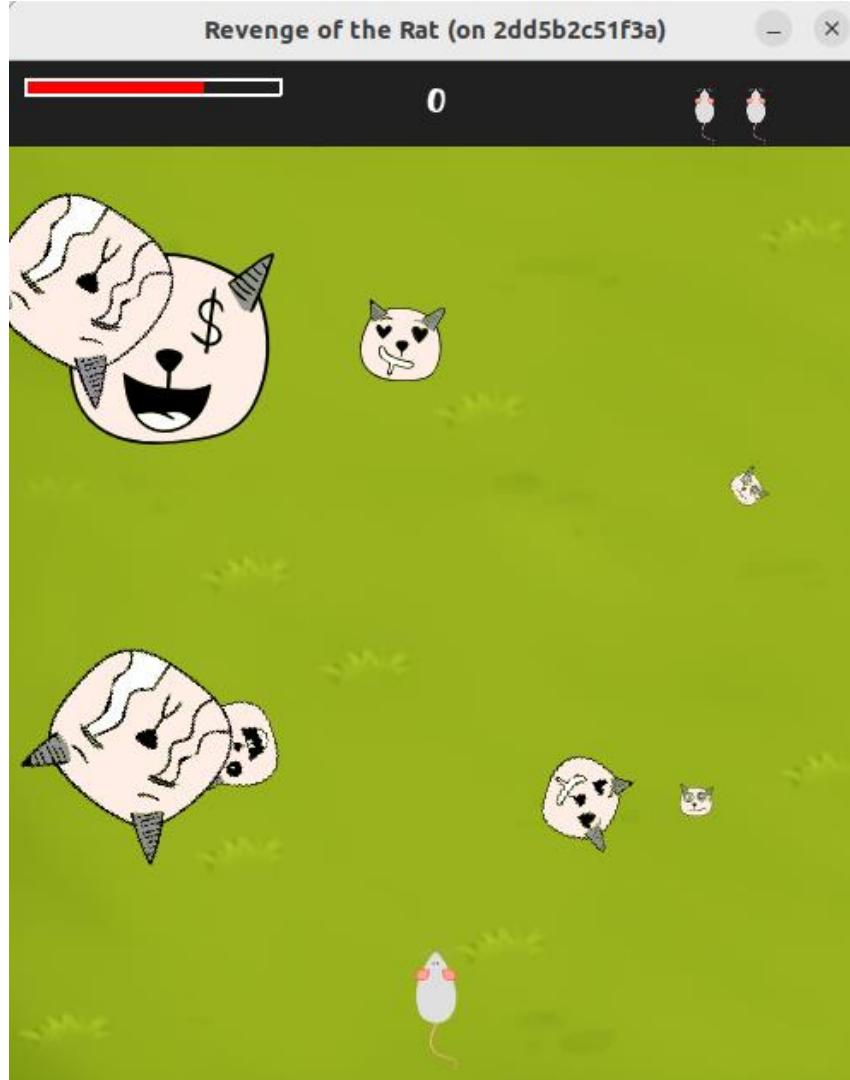
Cách sử dụng trò chơi:

Người chơi được quyền nhấn một phím bất kỳ trên bàn phím hoặc nhấn chuột để bắt đầu trò chơi. Sau đó người chơi được chọn một trong hai nhân vật “chuột” được thiết lập sẵn

trong trò chơi. Sau khi đã chọn được nhân vật yêu thích, trò chơi sẽ chính thức bắt đầu. Nhân vật “chuột” sẽ có nhiệm vụ tiêu diệt các nhân vật “mèo” bằng cách bắn vào chúng. Người chơi sử dụng cặp phím di chuyển trái và di chuyển phải hoặc phím A và D để di chuyển nhân vật “chuột”, sử dụng phím space hoặc nhấn chuột để thực hiện hành động bắn. Người chơi sẽ có ba lượt chơi tương ứng ba mạng của “chuột”, mỗi khi va vào “mèo” số lần nhất định, “chuột” sẽ mất một mạng và sau ba lần như vậy, trò chơi sẽ kết thúc.



Hình 2: Giao diện chọn nhân vật “chuột” của trò chơi



Hình 3: Giao diện chính của trò chơi

1.1.2. Sử dụng Docker cho ứng dụng

Tạo một file Dockerfile để tải cũng như cập nhật các thư viện cần thiết vào môi trường trong Docker để sử dụng cho trò chơi (pygame, xhost,...).

Một số câu lệnh đáng chú ý như sau:

- Dockerfile này mô tả cách xây dựng một hình ảnh Docker dựa trên Ubuntu 22.04.
- ENV DEBIAN_FRONTEND=noninteractive: Lệnh này thiết lập biến môi trường Debian_FRONTEND để chế độ không tương tác, giúp tránh việc hỏi các câu hình tương tác trong quá trình cài đặt.

- RUN apt-get update && apt-get upgrade -y && apt-get install -y ...: Lệnh này cập nhật danh sách gói và cài đặt một số gói phần mềm bao gồm thư viện và công cụ cần thiết cho ứng dụng.
- RUN apt install -qqy x11-apps: Lệnh này cài đặt ứng dụng x11-apps để hỗ trợ giao diện đồ họa.
- RUN pip3 install pygame: Lệnh này sử dụng pip3 để cài đặt thư viện Pygame.
- ARG USER=docker, ARG UID=1000, ARG GID=1000: Định nghĩa các tham số xây dựng (build argument) với giá trị mặc định.
- ARG PW=docker: Định nghĩa một mật khẩu mặc định cho người dùng.
- RUN useradd -m \${USER} --uid=\${UID} --shell /bin/bash && echo "\${USER}:\${PW}" | chpasswd \ && adduser docker sudo: Tạo một người dùng với tên là giá trị của USER (mặc định là "docker") và cài đặt mật khẩu cho người dùng này. Người dùng cũng được thêm vào nhóm sudo để có quyền thực thi các lệnh có đặc quyền.
- USER \${UID}:\${GID}: Chuyển đổi người dùng mặc định cho các lệnh tiếp theo trong Dockerfile. Điều này giúp giảm thiểu rủi ro bảo mật bằng cách chạy các lệnh với đặc quyền người dùng thấp hơn.
- WORKDIR /home/\${USER}: Đặt thư mục làm việc mặc định khi container được khởi chạy là thư mục home của người dùng (được định nghĩa bởi USER).

```

1 FROM ubuntu:20.04
2 ENV DEBIAN_FRONTEND=noninteractive
3
4 # install semua dependensi paket (library) yang dibutuhkan
5 RUN apt-get update && apt-get upgrade -y && apt-get install -y \
6     tzdata \
7     libssl-dev \
8     openssl \
9     zlib1g-dev \
10    build-essential \
11    checkinstall \
12    libffi-dev \
13    libsdl3-dev \
14    vim \
15    curl \
16    make \
17    sudo \
18    python3-pip \
19    python3-pygame \
20    libsdl1.2-dev \
21    libsdl-image1.2-dev \
22    libsdl-mixer1.2-dev \
23    libsdl-sound1.2-dev \
24    libsdl-ttf2.0-dev \
25    libsdl2-dev \
26    libsdl2-image-dev \
27    libsdl2-mixer-dev \
28    libsdl2-ttf-dev \
29    libsdl2-gfx-dev \
30    libsdl2-net-dev
31
32 # install x11
33 RUN apt install -qqy x11-apps
34
35 # install pygame
36 RUN pip3 install pygame
37
38 # menambahkan parameter build argument
39 ARG USER=docker
40 ARG UID=1000
41 ARG GID=1000
42
43 # menambahkan default password untuk user
44 ARG PW=docker
45 RUN useradd -m ${USER} --uid=${UID} --shell /bin/bash && echo "${USER}:$\
{PW}" | chpasswd \
46     && adduser docker sudo
47
48 # Setup default user, ketika memasuki kontainer
49 USER ${UID}:${GID}
50 WORKDIR /home/${USER}

```

Hình 4: Nội dung file Dockerfile

Tạo một file docker-compose.yml để build image và khởi chạy container. Đặt tên cho Docker image là “ratgame” và tên của Docker container là “rat_game”. Docker compose sẽ tạo một image dựa trên file Dockerfile.

Một số lệnh cơ bản trong docker-compose.yml:

- services: game: Tên của dịch vụ được định nghĩa là "game".
- image: game:latest: Sử dụng hình ảnh Docker có tên là "game" và phiên bản là "latest".
- container_name: Application_using_pygame: Đặt tên cho container là "Application_using_pygame".
- build: Xác định cách Docker sẽ xây dựng hình ảnh. Trong trường hợp này, sử dụng Dockerfile có tên là "Dockerfile-game" ở thư mục hiện tại (context).
- volumes: Liệt kê các kết nối giữa thư mục trên máy host và container.
- /tmp/.X11-unix:/tmp/.X11-unix: Chia sẻ thư mục X11 giữa máy host và container để hiển thị GUI.
- /run/dbus:/run/dbus: Chia sẻ thư mục D-Bus.
- /dev/shm:/dev/shm: Chia sẻ thư mục /dev/shm.
- /dev/snd:/dev/snd: Chia sẻ thư mục /dev/snd.
- /dev/dri:/dev/dri: Chia sẻ thư mục /dev/dri.
- \${XDG_RUNTIME_DIR}/pulse/native:\${XDG_RUNTIME_DIR}/pulse/native : Chia sẻ thư mục PulseAudio runtime.
- /run/user/1000/pulse:/run/user/1000/pulse: Chia sẻ thư mục PulseAudio runtime.
- /var/run/dbus:/var/run/dbus: Chia sẻ thư mục D-Bus runtime.
- /home/minhngoc/Downloads:/home/docker: Chia sẻ thư mục Downloads từ máy host vào /home/docker trong container.
- DISPLAY: Chỉ định biến môi trường DISPLAY, cần thiết để hiển thị GUI.
- PULSE_SERVER=unix:\${XDG_RUNTIME_DIR}/pulse/native: Đặt server PulseAudio cho container.
- XDG_RUNTIME_DIR=/run/user/1000: Đặt thư mục runtime XDG cho container.

```

1 version: '3'
2 services:
3   game:
4     image: game:latest
5     container_name: Application_using_pygame
6     build:
7       context: .
8       dockerfile: Dockerfile-game
9     privileged: true
10    stdin_open: true
11    tty: true
12    cap_add:
13      - SYS_PTRACE
14    user: "1000:1000"
15    volumes:
16      - /tmp/.X11-unix:/tmp/.X11-unix
17      - /run/dbus:/run/dbus
18      - /dev/shm:/dev/shm
19      - /dev/snd:/dev/snd
20      - /dev/dri:/dev/dri
21      - ${XDG_RUNTIME_DIR}/pulse/native:${XDG_RUNTIME_DIR}/pulse/native
22      - /run/user/1000/pulse:/run/user/1000/pulse
23      - /var/run/dbus:/var/run/dbus
24      - /home/minhngoc/Downloads:/home/docker
25    environment:
26      - DISPLAY
27      - PULSE_SERVER=unix:${XDG_RUNTIME_DIR}/pulse/native
28      - XDG_RUNTIME_DIR=/run/user/1000

```

Hình 5: Nội dung file docker-compose.yml

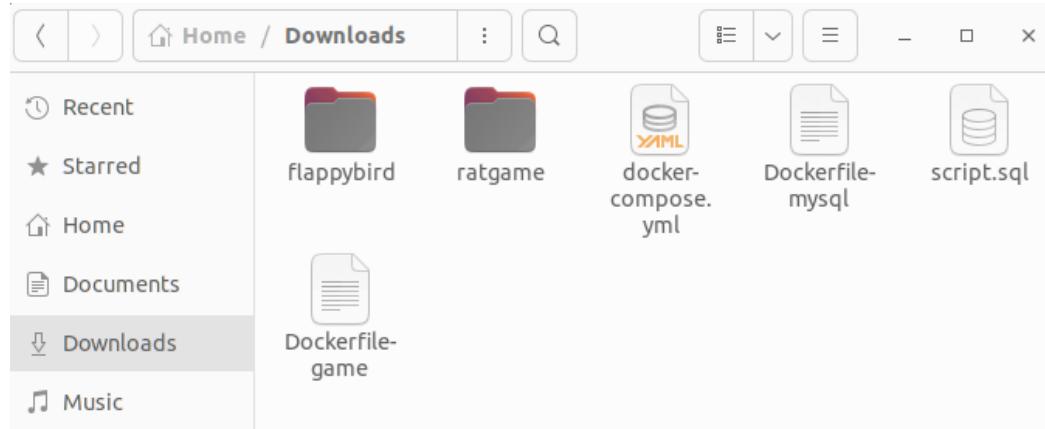
Để có thể sử dụng giao diện đồ họa của pygame và âm thanh trong Docker, chúng ta phải sử dụng một số lệnh docker run nhưng bởi vì các câu lệnh này khá dài và khó sử dụng cho người dùng nên thay vì phải tự chạy các lệnh rắc rối, chúng ta sẽ ánh xạ chúng trong mục Volumes của docker-compose.yml. Mục đích của mục Volumes là chạy các lệnh cho phép chia sẻ sử dụng giao diện đồ họa của pygame thông qua xhost và chia sẻ âm thanh của máy chủ sang môi trường Docker. Docker sẽ mount các file hiện có trong thư mục làm việc hiện tại của máy chủ sang thư mục /home/docker và bây giờ thì người dùng có toàn quyền thực thi đối với các file này. (lưu ý: thay đổi thư mục hiện hành để có thể mount file thành công)

1.1.3. Hướng dẫn cài đặt

Tải các file về máy tính. Sử dụng đường dẫn:

https://github.com/MN911718/NT132.O11.ANTT_Project.git

Các file được tải xuống sẽ như hình bên dưới:



Hình 6: Các file và thư mục của trò chơi

Chỉnh đường dẫn file trong docker-compose.yml: thực hiện thay đổi đường dẫn này thành đường dẫn đang lưu các file đã tải về (VD: /home[minhngoc]/Downloads là thư mục đang lưu trữ các file ở trên)

volumes:

```

- /tmp/.X11-unix:/tmp/.X11-unix
- /run/dbus:/run/dbus
- /dev/shm:/dev/shm
- /dev/snd:/dev/snd
- /dev/dri:/dev/dri
- ${XDG_RUNTIME_DIR}/pulse/native:${XDG_RUNTIME_DIR}/pulse/native
- /run/user/1000/pulse:/run/user/1000/pulse
- /var/run/dbus:/var/run/dbus
- /home/minhngoc/Downloads:/home/docker

```

Hình 7: Thay đổi đường dẫn

Thực hiện chạy docker-compose:

```
$sudo docker-compose build
$sudo docker-compose up
```

```

minhngoc@minhngoc-virtual-machine:~/Downloads$ sudo docker-compose build
WARNING: The XDG_RUNTIME_DIR variable is not set. Defaulting to a blank string.
Building game
[+] Building 4.0s (10/10) FINISHED                                            docker:default
  => [internal] load build definition from Dockerfile-game                  0.0s
  => => transferring dockerfile: 1.12kB                                     0.0s
  => [internal] load .dockerignore                                         0.0s
  => => transferring context: 2B                                           0.0s
  => [internal] load metadata for docker.io/library/ubuntu:22.04           4.0s
  => [1/6] FROM docker.io/library/ubuntu:22.04@sha256:6042500cf4b44023ea18 0.0s
  => CACHED [2/6] RUN apt-get update && apt-get upgrade -y && apt-get inst 0.0s
  => CACHED [3/6] RUN apt install -qqy x11-apps                           0.0s
  => CACHED [4/6] RUN pip3 install pygame                                    0.0s
  => CACHED [5/6] RUN useradd -m docker --uid=1000 --shell /bin/bash && ec 0.0s
  => CACHED [6/6] WORKDIR /home/docker                                     0.0s
  => exporting to image                                                 0.0s
  => => exporting layers                                              0.0s
  => => writing image sha256:e1abb94c9544b3250ba82fc6886f90b083080cdad9f 0.0s
  => => naming to docker.io/library/game:latest                           0.0s
Building build-sql
[+] Building 0.1s (7/7) FINISHED                                            docker:default
  => [internal] load build definition from Dockerfile-mysql                0.0s
  => => transferring dockerfile: 219B                                      0.0s
  => [internal] load .dockerignore                                         0.0s
  => => transferring context: 2B                                           0.0s
  => [internal] load metadata for docker.io/library/mysql:latest          0.0s
  => [internal] load build context                                       0.0s
  => => transferring context: 32B                                         0.0s
  => [1/2] FROM docker.io/library/mysql:latest                           0.0s
  => CACHED [2/2] ADD script.sql /docker-entrypoint-initdb.d             0.0s
  => exporting to image                                                 0.0s
  => => exporting layers                                              0.0s
  => => writing image sha256:fd868a87fa732aaae41117eb768aa0746bead3d1cd470 0.0s
  => => naming to docker.io/library/build-sql:latest                      0.0s
minhngoc@minhngoc-virtual-machine:~/Downloads$ sudo docker-compose up
WARNING: The XDG_RUNTIME_DIR variable is not set. Defaulting to a blank string.
Starting Application_using_mysql ... done
Starting Application_using_pygame ... done
Attaching to Application_using_mysql, Application_using_pygame
Application_using_mysql | 2024-01-05 11:17:39+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
Application_using_mysql | 2024-01-05 11:17:40+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'

```

Hình 8: Chạy file docker-compose

Sau khi file docker-compose đã chạy thành công, dùng lệnh \$docker ps để kiểm tra container.

```

minhngoc@minhngoc-virtual-machine:~/Downloads$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND       CREATED        STATUS
              PORTS
5238409fb09d   game:latest    "/bin/bash"   10 days ago   Up About a minute
191e265e2509   build-sql:latest "docker-entrypoint.s..." 10 days ago   Up About a minute
              0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp
minhngoc@minhngoc-virtual-machine:~/Downloads$
```

Hình 9: Kiểm tra container

Thực hiện truy cập vào Application_using_pygame và kiểm tra xem docker đã được ánh xạ thành công chưa.

```

minhngoc@minhngoc-virtual-machine:~/Downloads$ sudo docker exec -it 5238409fb09d /bin/bash
[sudo] password for minhngoc:
docker@5238409fb09d:~$ ls
Dockerfile-game Dockerfile-mysql docker-compose.yml flappybird ratgame script.sql
docker@5238409fb09d:~$
```

Hình 10: Kiểm tra file

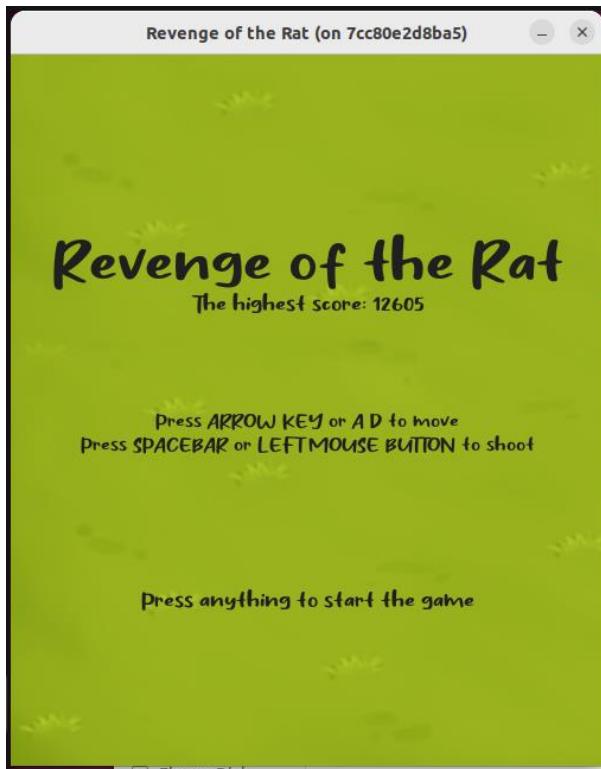
Bây giờ chạy lệnh python3 main.py để thực thi file game:

```

minhngoc@minhngoc-virtual-machine:~/Downloads$ sudo docker exec -it 5238409fb09d /bin/bash
[sudo] password for minhngoc:
docker@5238409fb09d:~$ ls
Dockerfile-game Dockerfile-mysql docker-compose.yml flappybird ratgame script.sql
docker@5238409fb09d:~$ cd ratgame
docker@5238409fb09d:~/ratgame$ ls
Data LICENSE.md README.md __pycache__ font.otf head.png img main.py sound
docker@5238409fb09d:~/ratgame$ python3 main.py
pygame 2.1.2 (SDL 2.0.20, Python 3.10.12)
Hello from the pygame community. https://www.pygame.org/contribute.html

```

Hình 11: Truy cập file ratgame



Hình 12: Kết quả chạy trò chơi thành công

1.2. Flappybird

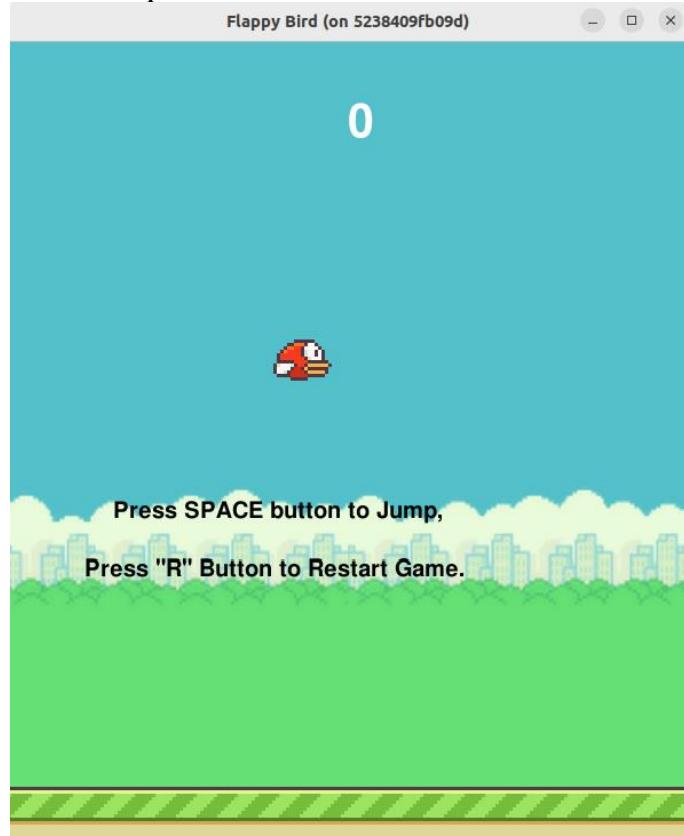
1.2.1. Giới thiệu về ứng dụng

Flappy Bird là một trò chơi điện tử đơn giản rất quen thuộc đối với những tín đồ chơi game điện tử bởi tính thú vị và cực kỳ gây nghiện. Người chơi điều khiển một chú chim nhỏ, vượt qua các ống đường ống nằm ngang bằng cách nhấn phím để nhảy lên. Đơn giản về cách chơi nhưng Flappy Bird nổi tiếng với độ khó cao, tạo ra thách thức và sự cạm bẫy, khiến cho người chơi muốn vượt qua mức điểm cao của mình. Trò chơi có sử dụng giao diện đồ họa và âm thanh.

Cách sử dụng trò chơi:

Người chơi sử dụng phím space trên bàn phím để bắt đầu trò chơi. Cách chơi vô cùng đơn giản, người chơi chỉ cần dùng phím space để điều khiển chú chim bay lượn và vượt qua các đường ống nằm ngang rải rác trên đường đi. Hệ thống cho phép ghi lại số

điểm của người chơi tương ứng với số đường ống người chơi đã vượt qua. Khi người chơi thua cuộc, chỉ cần nhấn phím R để bắt đầu lại.



Hình 13: Giao diện bắt đầu của trò chơi

1.2.2. Sử dụng Docker cho ứng dụng

Bởi vì trò chơi này cũng sử dụng các thư viện như pygame, X11 như ứng dụng trò chơi Revenger of the Rat nên file Dockerfile được sử dụng chung.

Chi tiết file Dockerfile được viết ở mục **1.1.2**.

1.2.3. Hướng dẫn cài đặt

File trò chơi Flappybird nằm chung với trò chơi Revenger of the Rat. Thực hiện các bước như phần **1.1.3**. Sau khi chạy file docker-compose, ta tiến hành truy cập vào thư mục flappybird và sử dụng lệnh \$python3 main.py để bắt đầu chơi game.

```
docker@5238409fb09d:~$ ls
Dockerfile-game Dockerfile-mysql docker-compose.yml flappybird ratgame script.sql
docker@5238409fb09d:~$ cd flappybird
docker@5238409fb09d:~/flappybird$ ls
__pycache__ assets bird.py game.py main.py pipe.py settings.py sound world.py
docker@5238409fb09d:~/flappybird$ python3 main.py
pygame 2.1.2 (SDL 2.0.20, Python 3.10.12)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

Hình 14: Chạy file main.py

2. Ứng dụng MySQL

2.1. Giới thiệu về ứng dụng

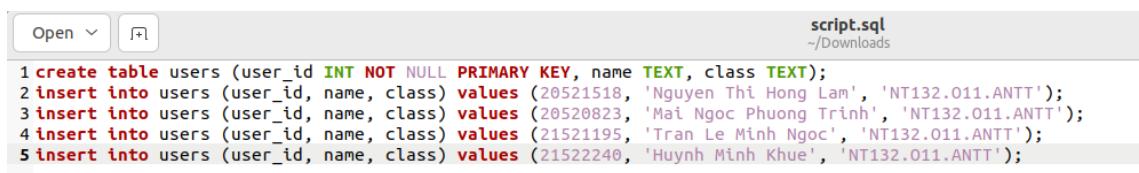
MySQL là một hệ quản trị cơ sở dữ liệu phổ biến, mã nguồn mở, được sử dụng rộng rãi trong phát triển ứng dụng web và nhiều dự án khác. Với khả năng quản lý dữ liệu mạnh mẽ, MySQL cung cấp một môi trường linh hoạt và hiệu quả để lưu trữ và truy xuất dữ liệu.

Khi tích hợp MySQL vào môi trường Docker, người phát triển có thể hưởng lợi từ những đặc tính linh hoạt và di động của container. Việc sử dụng MySQL trên Docker mang lại nhiều lợi ích, bao gồm:

- **Dễ dàng triển khai và di động:** Docker cho phép triển khai MySQL một cách nhanh chóng và có thể di chuyển giữa các môi trường mà không gặp vấn đề tương thích. Điều này giúp giảm bớt khó khăn khi chuyển đổi giữa các môi trường phát triển, thử nghiệm và sản xuất.
- **Tích hợp linh hoạt:** MySQL trên Docker có thể tích hợp dễ dàng với các ứng dụng và dịch vụ khác trong môi trường container. Việc quản lý các dependency và môi trường làm việc trở nên thuận tiện hơn.
- **Quản lý tài nguyên hiệu quả:** Docker giúp kiểm soát và quản lý tài nguyên của MySQL một cách hiệu quả, giúp tối ưu hóa việc sử dụng bộ nhớ và CPU, cũng như thuận lợi trong việc mở rộng dữ liệu và hiệu suất hệ thống.
- **Bảo mật và cô lập:** Sử dụng MySQL trong container giúp tăng cường bảo mật và cô lập. Các container có thể chạy trong môi trường đóng và không tác động đến các thành phần khác trên hệ thống.

2.2. Sử dụng Docker cho ứng dụng

Tạo một file sql đơn giản trên máy chủ. Ở đây nhóm tạo một file sql chứa thông tin của các thành viên trong nhóm và đặt tên là script.sql.



The screenshot shows a code editor window with the following content:

```
Open [ ] script.sql ~ /Downloads
1 create table users (user_id INT NOT NULL PRIMARY KEY, name TEXT, class TEXT);
2 insert into users (user_id, name, class) values (20521518, 'Nguyen Thi Hong Lam', 'NT132.011.ANTT');
3 insert into users (user_id, name, class) values (20520823, 'Mai Ngoc Phuong Trinh', 'NT132.011.ANTT');
4 insert into users (user_id, name, class) values (21521195, 'Tran Le Minh Ngoc', 'NT132.011.ANTT');
5 insert into users (user_id, name, class) values (21522240, 'Huynh Minh Khue', 'NT132.011.ANTT');
```

Hình 15: File script.sql

Thực hiện tạo một Dockerfile để cấu hình như sau:

- **FROM mysql:latest:** Dòng này xác định image cơ sở cho container. Trong trường hợp này, image được chọn là mysql:latest, có nghĩa là sử dụng phiên bản mới nhất của MySQL từ Docker Hub.

- ENV MYSQL_ROOT_PASSWORD=123: Thiết lập biến môi trường trong container để xác định mật khẩu của tài khoản root của MySQL. Trong ví dụ này, mật khẩu được thiết lập là "123".
- ENV MYSQL_DATABASE users: Thiết lập biến môi trường để xác định tên của cơ sở dữ liệu mà bạn muốn tạo. Trong ví dụ này, cơ sở dữ liệu được tạo có tên là "users".
- ENV MYSQL_USER minhngoc: Thiết lập biến môi trường để xác định tên người dùng của MySQL. Trong ví dụ này, người dùng được tạo có tên là "minhngoc".
- ENV MYSQL_PASSWORD 1234: Thiết lập biến môi trường để xác định mật khẩu cho người dùng MySQL. Trong ví dụ này, mật khẩu được thiết lập là "1234".
- ADD script.sql /docker-entrypoint-initdb.d: Chép một tệp "script.sql" từ thư mục hiện tại của Dockerfile vào thư mục /docker-entrypoint-initdb.d trong container. Các tệp SQL trong thư mục này sẽ được tự động thực thi khi container được khởi động, giúp khởi tạo cơ sở dữ liệu.
- EXPOSE 3306: Khai báo rằng container sẽ lắng nghe trên cổng 3306, đây là cổng sử dụng cho mysql.

```

1 FROM mysql:latest
2 ENV MYSQL_ROOT_PASSWORD=123
3 ENV MYSQL_DATABASE users
4 ENV MYSQL_USER minhngoc
5 ENV MYSQL_PASSWORD 1234
6 ADD script.sql /docker-entrypoint-initdb.d
7 EXPOSE 3306

```

Hình 16: Cấu hình Dockerfile

Tạo một file docker-compose.yml. Trong trường hợp này, docker-compose được tạo chung với **1. Ứng dụng trò chơi pygame**.

Dưới đây là chi tiết cho ứng dụng mysql:

- image: build-sql:latest: Xác định image sẽ được sử dụng cho dịch vụ MySQL. Trong trường hợp này, image có tên là "build-sql" và sử dụng tag "latest".
- container_name: Application_using_mysql: Đặt tên cho container được tạo ra từ image. Trong trường hợp này, container được đặt tên là "Application_using_mysql".
- build: Cung cấp các thông tin để xây dựng image của MySQL từ Dockerfile. dockerfile: Dockerfile-mysql: Định rõ tên của Dockerfile mà Docker nên sử dụng để xây dựng image.

- ports: - "3306:3306": Chuyển tiếp cổng của container đến cổng của máy host. Trong trường hợp này, cổng 3306 của container MySQL sẽ được chuyển tiếp đến cổng 3306 của máy host, cho phép ứng dụng khác truy cập MySQL qua cổng này.

```
build-sql:
  image: build-sql:latest
  container_name: Application_using_mysql
  build:
    context: .
    dockerfile: Dockerfile-mysql
  ports:
    - "3306:3306"
```

Hình 17: Cấu hình docker-compose.yml

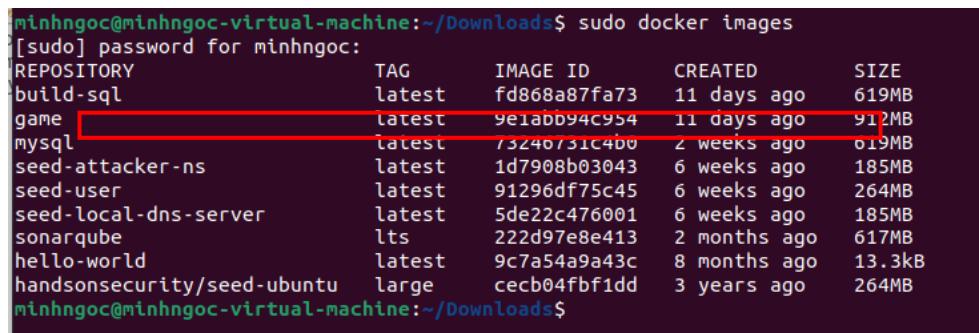
2.3. Hướng dẫn cài đặt

Download container mysql sẵn có trên Docker về bằng lệnh

\$docker pull mysql

Kiểm tra xem mysql đã được tải xuống thành công chưa bằng lệnh

\$sudo docker images



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
build-sql	latest	fd868a87fa73	11 days ago	619MB
game	latest	9e1add94c954	11 days ago	912MB
mysql	latest	73246731c4b0	2 weeks ago	619MB
seed-attacker-ns	latest	1d7908b03043	6 weeks ago	185MB
seed-user	latest	91296df75c45	6 weeks ago	264MB
seed-local-dns-server	latest	5de22c476001	6 weeks ago	185MB
sonarqube	lts	222d97e8e413	2 months ago	617MB
hello-world	latest	9c7a54a9a43c	8 months ago	13.3kB
handsonsecurity/seed-ubuntu	large	cecb04fbf1dd	3 years ago	264MB

Hình 18: Kiểm tra images

Thực hiện build và khởi động docker-compose như phần 1.1.3. Sau đó truy cập vào mysql, sử dụng lệnh

\$sudo docker exec -it 191e265e2509 /bin/bash

Sử dụng tài khoản đã tạo trong Dockerfile để truy cập vào cơ sở dữ liệu.



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	NAMES
5238409fb09d	game:latest	"/bin/bash"	10 days ago	Up About a minute	Application_using_pygame
191e265e2509	build-sql:latest	"docker-entrypoint.s..."	10 days ago	Up About a minute	Application_using_mysql
		0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp			

Hình 19: Kiểm tra container

```

minhngoc@minhngoc-virtual-machine:~/Downloads$ sudo docker exec -it 191e265e2509 /bin/bash
bash-4.4# mysql -uminhngoc -p1234
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```

Hình 20: Truy cập container mysql

Sử dụng lệnh \$show database;, ta có thể thấy đã có cơ sở dữ liệu tên “user” lúc nãy chúng ta vừa tạo.

```

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| performance_schema |
| users |
+-----+
3 rows in set (0.00 sec)

```

Hình 21: Kiểm tra database

Dùng lệnh \$select * from users; để xem thông tin trong script.sql đã được ánh xạ thành công hay chưa.

```

mysql> select * from users;
+-----+-----+-----+
| user_id | name          | class   |
+-----+-----+-----+
| 20520823 | Mai Ngoc Phuong Trinh | NT132.011.ANTT |
| 20521518 | Nguyen Thi Hong Lam | NT132.011.ANTT |
| 21521195 | Tran Le Minh Ngoc | NT132.011.ANTT |
| 21522240 | Huynh Minh Khue | NT132.011.ANTT |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

Hình 22: Xem các trường thông tin trong users

Trong sql này, ta có thể thực hiện các thao tác như thêm, xóa, sửa một cách nhanh chóng.

```

mysql> insert into users (user_id, name, class) values (12345678, 'minh ngoc', 'NT132.011.ANTT');
Query OK, 1 row affected (0.01 sec)

mysql> select * from users;
+-----+-----+-----+
| user_id | name          | class   |
+-----+-----+-----+
| 12345678 | minh ngoc      | NT132.011.ANTT |
| 20520823 | Mai Ngoc Phuong Trinh | NT132.011.ANTT |
| 20521518 | Nguyen Thi Hong Lam | NT132.011.ANTT |
| 21521195 | Tran Le Minh Ngoc | NT132.011.ANTT |
| 21522240 | Huynh Minh Khue | NT132.011.ANTT |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

Hình 23: Thêm dữ liệu

```

mysql> delete from users where user_id = 12345678;
Query OK, 1 row affected (0.00 sec)

mysql> select * from users;
+-----+-----+-----+
| user_id | name      | class    |
+-----+-----+-----+
| 20520823 | Mai Ngoc Phuong Trinh | NT132.011.ANTT |
| 20521518 | Nguyen Thi Hong Lam   | NT132.011.ANTT |
| 21521195 | Tran Le Minh Ngoc    | NT132.011.ANTT |
| 21522240 | Huynh Minh Khue       | NT132.011.ANTT |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

Hình 24: Xóa dữ liệu

3. Ứng dụng Blog cá nhân

3.1. Giới thiệu về ứng dụng

a. Sơ lược về Blog

Blog về món ăn là một nơi tuyệt vời để chia sẻ đam mê nấu ăn và thưởng thức đồ ăn ngon. Nơi đây chúng tôi không chỉ chia sẻ cách nấu ăn mà còn mời bạn tham gia vào hành trình khám phá thế giới ngon lành và thú vị của ẩm thực. Tại không gian chung này, chúng tôi không chỉ là những đầu bếp đam mê nấu ăn, mà còn là những người yêu thưởng thức mỗi hương vị và câu chuyện đằng sau từng bữa ăn. Blog sẽ giới thiệu đến mọi người các món ăn hấp dẫn mà bản thân đã được trải nghiệm. Chia sẻ những hương vị tuyệt vời của nền ẩm thực đa dạng và phong phú trên toàn thế giới. Chúng tôi sẽ dẫn dắt bạn qua những hành trình mua sắm thú vị để lựa chọn những nguyên liệu tươi ngon nhất, cũng như chia sẻ bí quyết để tạo ra những món ăn độc đáo và hấp dẫn và tham gia vào cuộc phiêu lưu của chúng tôi, khám phá các điểm ẩm thực độc đáo, đánh giá nhà hàng và chia sẻ những câu chuyện về hương vị.

b. Tại sao lại xây dựng blog trên docker?

Xây dựng một Blog chia sẻ các món ăn chạy trên Docker có thể mang lại nhiều lợi ích và tiện ích cho người quản lý và người sử dụng.

- Docker cho phép đóng gói ứng dụng cùng với tất cả các phụ thuộc của nó vào một container.
- Docker cung cấp môi trường đóng gói nhẹ và dễ quản lý.
- Containers cung cấp mức độ cô lập giữa các ứng dụng, giúp ngăn chặn tác động tiêu cực từ một ứng dụng đến các ứng dụng khác.
- Docker giúp dễ dàng mở rộng ứng dụng của người dùng bằng cách triển khai nhiều container chia sẻ tài nguyên hệ thống.

3.3. Sử dụng Docker cho ứng dụng

Sau đây là các bước thực hiện để tạo trang Blog chạy trên docker:

- Tạo cơ sở dữ liệu trên phpMyAdmin gồm:

- Bảng user: lưu trữ tên đăng nhập và password của admin để đăng nhập và có quyền thêm hình ảnh vào Blog.
- Bảng foods: Lưu trữ tên món ăn và link url hình ảnh của món ăn.

The screenshot shows two separate instances of the phpMyAdmin interface. Both instances have the same sidebar navigation:

- Mới dùng
- Ua dùng
- information_schema
- performance_schema
- php_docker
 - Mới
 - foods
 - user

Top Instance (Bảng: foods):

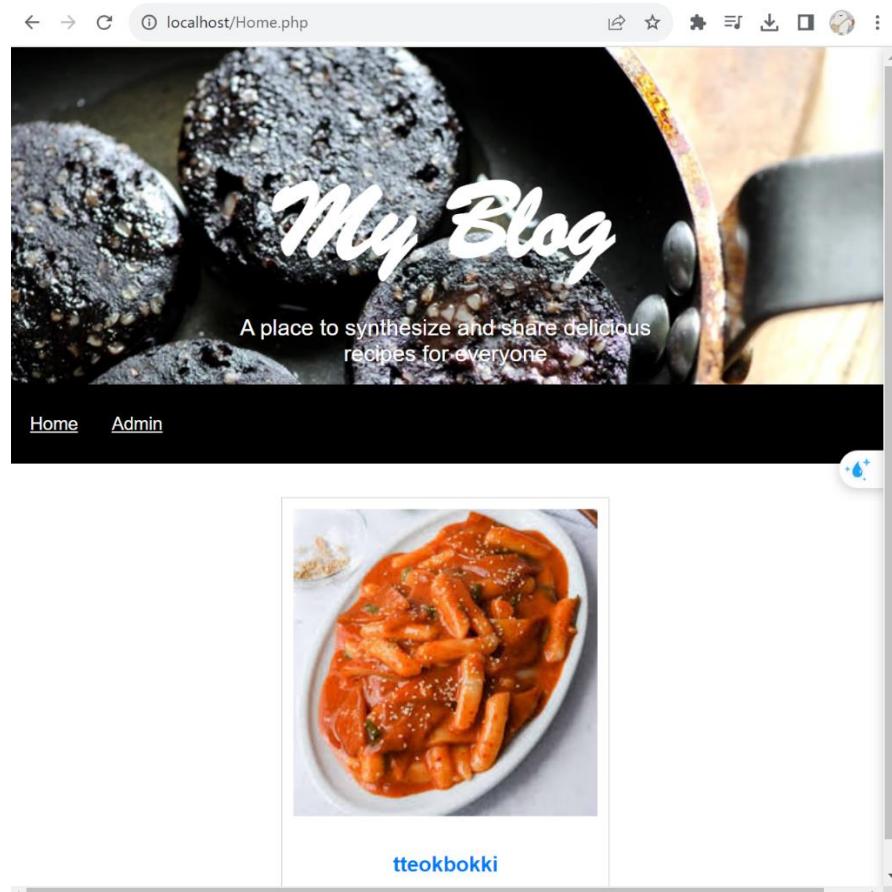
#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định	C
1	title	varchar(255)	utf8mb4_0900_ai_ci		Không	Không	c
2	image_url	varchar(1000)	utf8mb4_0900_ai_ci		Không	Không	c

Bottom Instance (Bảng: user):

#	Tên	Kiểu	Bảng mã đối chiếu	Thuộc tính	Null	Mặc định	G
1	username	varchar(255)	utf8mb4_0900_ai_ci		Không	Không	c
2	password	varchar(255)	utf8mb4_0900_ai_ci		Không	Không	c

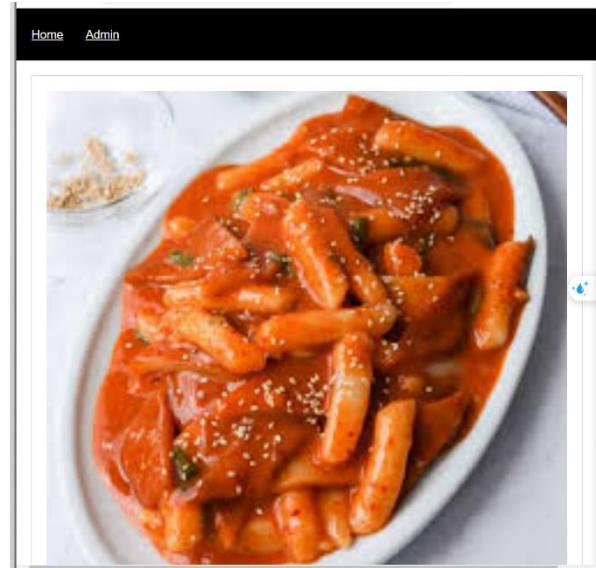
Hình 25: Database của Blog

- Tạo giao diện các trang web
 - Trang chủ của Blog



Hình 26: Trang chủ của Blog

- Chi tiết món ăn



Hình 27: Chi tiết một món ăn

- Trang đăng nhập với quyền admin

The screenshot shows a login form titled "Đăng Nhập" (Login). It contains two input fields: "Tên đăng nhập" (Username) and "Mật khẩu" (Password), both with placeholder text. Below the fields is a large blue button labeled "Đăng Nhập" (Login). At the top left of the page, there are navigation links "Home" and "Admin". A small circular icon with a blue plus sign is visible on the right side.

Hình 28: Trang admin

- Thêm món ăn

The screenshot shows a form titled "ADD FOOD". It has two input fields: "Food Title" and "Food Image URL", followed by a blue "Add Food" button. At the top left, there are navigation links "Home" and "Admin".

Hình 29: Trang thêm món ăn

- Tạo file docker compose để chạy web. Build và chạy trang web trên docker

```
docker-compose.yml
 1  version: '3.8'
 2  services:
 3    db:
 4      image: mysql:latest
 5      environment:
 6        - MYSQL_DATABASE=php_docker
 7        - MYSQL_USER=php_docker
 8        - MYSQL_PASSWORD=password
 9        - MYSQL_ALLOW_EMPTY_PASSWORD=1
10      volumes:
11        - "./db:/docker-entrypoint-initdb.d"
12    www:
13      image: php:apache
14      volumes:
15        - "./:/var/www/html"
16      ports:
17        - 80:80
18        - 443:443
19    phpmyadmin:
20      image: phpmyadmin/phpmyadmin
21      ports:
22        - 8001:80
23      environment:
24        - PMA_HOST=db
25        - PMA_PORT=3306
```

Hình 30: Docker-compose.yml

3.4. Hướng dẫn cài đặt

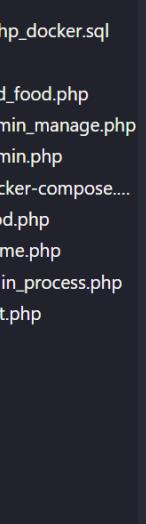
Môi trường, thiết bị thực hành: windows

Cài đặt phần mềm Docker, VScode

Down load folder theo đường dẫn:

<https://github.com/PTrinh116/QuanTriMang.git>

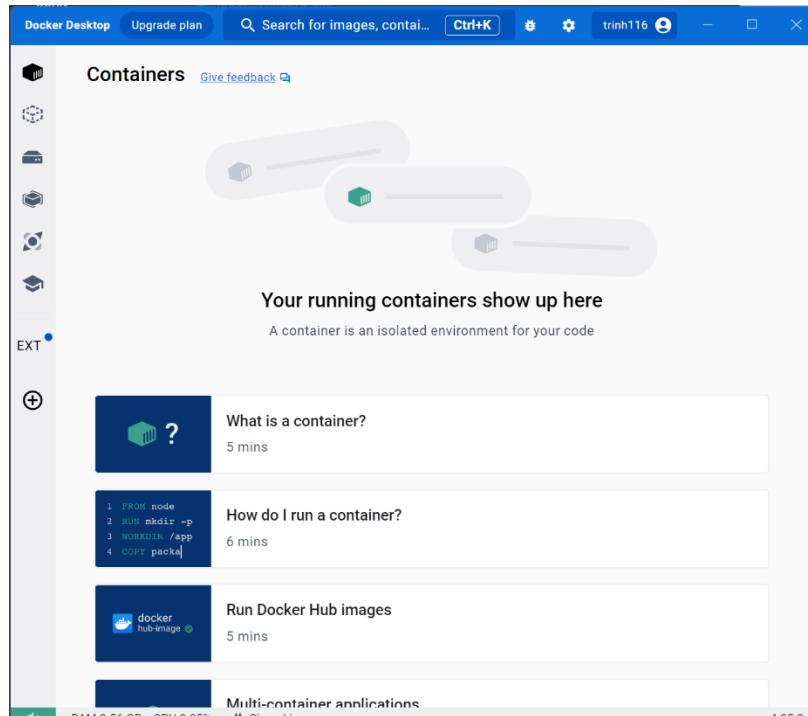
- Mở folder trên vscode.



```
version: '3.8' # Adjust the version according to your Docker
# Compose version
services:
  db:
    image: mysql:latest
    environment:
      - MYSQL_DATABASE=php_docker
      - MYSQL_USER=php_docker
      - MYSQL_PASSWORD=password # this should live in a env
      - MYSQL_ALLOW_EMPTY_PASSWORD=1 # equivalent to True
    volumes:
      - "./db:/docker-entrypoint-initdb.d" # this is how we
        # sync the current dir on local to container
  www:
    image: php:apache
    volumes:
      - ".:/var/www/html" # sync the current dir on local to
        # container
    ports:
      - 80:80
      - 443:443 # for future ssl traffic
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    ports:
```

Hình 31: Mở folder trên VSCode

- Mở Docker



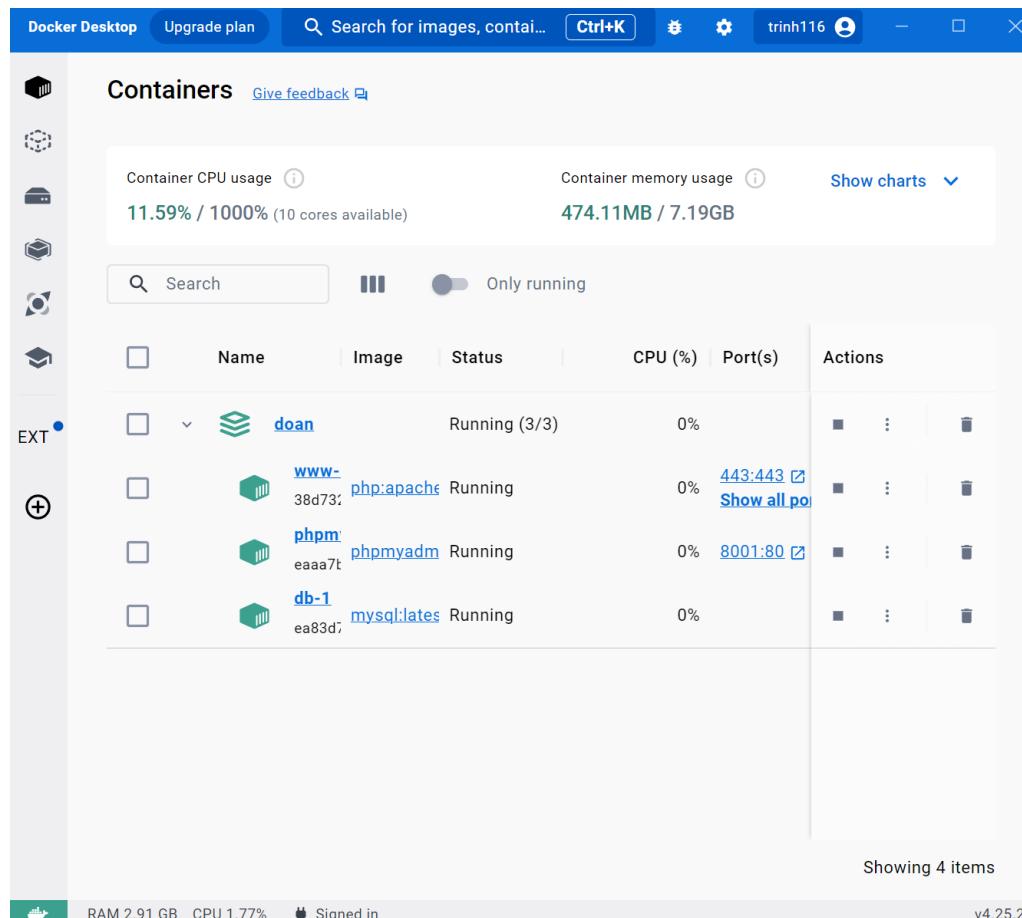
Hình 32: Mở docker

- Chạy lệnh: docker-compose up

```
ad2afdb99a9d: Download complete
dbc5aa907229: Downloading [>]
82f252ab4ad1: Waiting
bf5b34fc9894: Waiting
6161651d3d95: Waiting
cf2adf296ef1: Waiting
29cb0e922a81: Waiting
1d8316909cff: Waiting
de56d47ac89d: Waiting
cb61e9d4ce3c: Waiting
85e8b0392368: Waiting
a209c4be7efc: Waiting
7a6db449b51b: Downloading [>]
```

Hình 33: Chạy lệnh docker-compose up

- Đợi build thành công



Hình 34: Quá trình build

- Truy cập vào localhost:8001 để xem cơ sở dữ liệu

Mô hình | Ua dùng

information_schema
performance_schema
php_docker
Món
foods
user

Máy chủ: db:3306 > Cơ sở dữ liệu: php_docker > Bảng: foods

Duyệt Cấu trúc SQL Tìm kiếm Chèn Xuất Nhập Thao tác Bảy

Cấu trúc bảng Hiển thị quan hệ

#	Tên	Kiểu	Bảng mã dài chiều	Thuộc tính	Null	Mặc định	Ghi chú	Thêm	Hành động
1	title	varchar(255)	utf8mb4_0900_ai_ci		Không	Không		Thay đổi Xóa Thêm	Thay đổi Xóa Thêm
2	image_url	varchar(1000)	utf8mb4_0900_ai_ci		Không	Không		Thay đổi Xóa Thêm	Thay đổi Xóa Thêm

Theo dõi bảng Lưu mục đã chọn Duyệt Thay đổi Xóa Chính Duy nhất Chỉ mục Spatial Toán văn

In Di chuyển các cột Thêm 1 cột sau image_url Thực hiện

Chỉ mục

Hành động	Tên khóa	Kiểu	Duy nhất	Đã đóng gói	Cột	Số lượng	Bảng mã dài chiều	Null	Chú thích
Sửa Xóa	PRIMARY	BTREE	Có	Không	title	0	A	Không	

Create an index on 1 columns Thực hiện

Phân vùng

Chưa định nghĩa phân vùng nào!

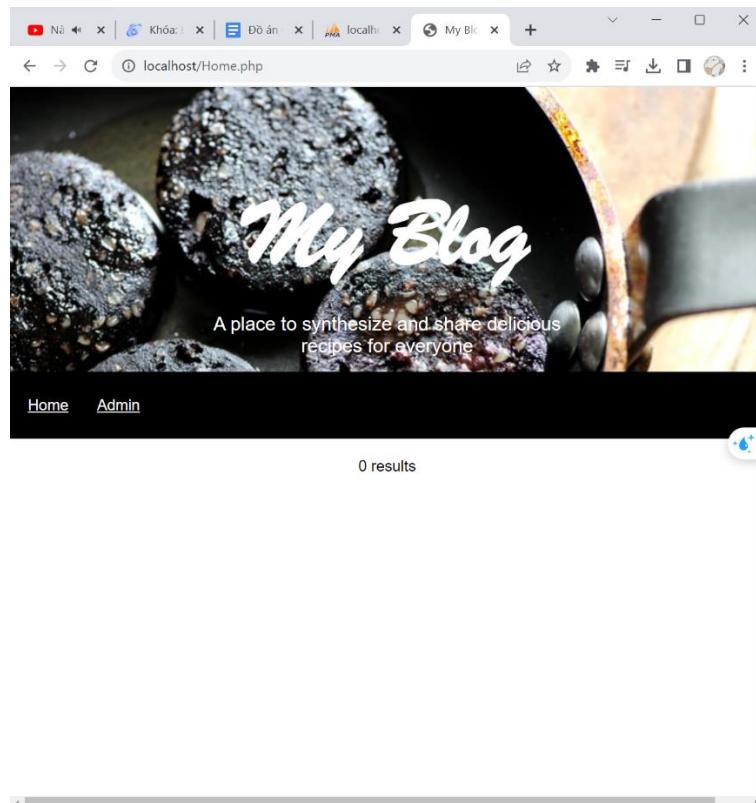
Bảng phân vùng

Thông tin

Dung lượng đĩa đã dùng	Thông kê dòng
Đĩa trống	đóng
Bảng đầu khuyến	15,0 KB

Hình 35: Truy cập vào localhost:8001

- Truy cập vào localhost:Home.php để vào Blog



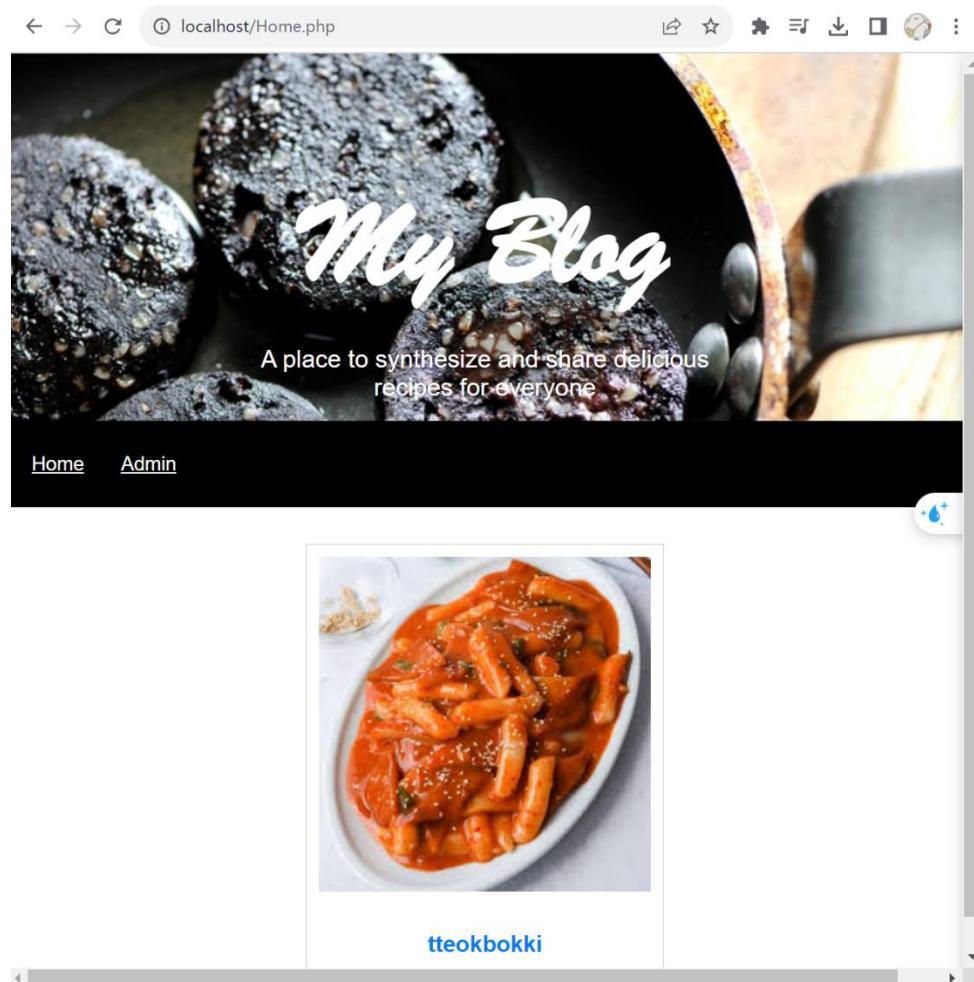
Hình 36: Trang chủ của Blog

- Để thêm món ăn vào Blog: chọn Admin -> đăng nhập -> thêm tên và link hình ảnh

The screenshot displays two pages of a web application:

- Login Page:** A modal window titled "Đăng Nhập" (Login) is shown. It contains two input fields: one for "tên" (name) with the value "trinh" and another for ".....". Below the fields is a blue "Đăng Nhập" (Login) button. The background of the page shows a navigation bar with "Home" and "Admin" links.
- Add Food Page:** A second modal window titled "ADD FOOD" is shown. It contains three input fields: one for "Food Name" with the value "tteokbokki", one for "Food Description" with the value "\{EZAomVGZNg&usqp=CAU", and a third for "Add Food". The background shows the same navigation bar.

Hình 37: Trang thêm món ăn mới



Hình 38: Đã thêm món ăn thành công

3.5. Công việc tương lai

- Áp dụng các biện pháp bảo mật tốt nhất cho Docker containers, bao gồm giảm đặc quyền, kiểm soát kết nối mạng.
- Phát triển thêm chức năng như bình luận, đánh giá món ăn, đăng nhập người dùng để tạo ra một trải nghiệm người dùng đầy đủ.
- Sử dụng công cụ như Docker Compose profiles để quản lý cấu hình dữ liệu và hình ảnh cho môi trường phát triển và triển khai.

4. Ứng dụng ghi chú Todolist

4.1. Giới thiệu về ứng dụng

Ứng dụng Todolist - trợ thủ đắc lực giúp người dùng quản lý công việc một cách hiệu quả và tổ chức cuộc sống hàng ngày của mình. Với giao diện đơn giản nhưng mạnh mẽ,

Todolist không chỉ là ứng dụng ghi chú mà còn là đối tác đồng hành đáng tin cậy để đưa ra lịch trình và hoàn thành mục tiêu.

Ứng dụng giúp người dùng dễ dàng tạo, sắp xếp và theo dõi danh sách công việc hàng ngày. Người dùng có thể nhanh chóng thêm công việc mới, thiết lập deadline, và đánh dấu công việc đã hoàn thành để theo dõi tiến độ.

Tham khảo:

https://github.com/ThanhLa1802/web_framework/tree/master/FLASK_PROJECT/Todolist_Tutorial/todolist

Mô tả ứng dụng:

Ứng dụng sử dụng thư viện flask và SQL.

Có thanh header để chuyển qua lại giữa các trang.

- Đăng nhập (Login):
 - *Tính năng*: Để có thể sử dụng app, người dùng bắt buộc phải đăng nhập. Nếu nhập sai mật khẩu hoặc email chưa được đăng ký, cảnh báo sẽ hiện lên. Ngược lại thì thông báo đăng nhập thành công. Sau khi đăng nhập thành công, người dùng sẽ được chuyển tới trang “Home” - nơi lưu trữ và cho phép người dùng thêm văn bản. App sẽ lưu lại trạng thái của lần đăng nhập trước đó.
 - *Giao diện*: Trang đăng nhập bao gồm khung nhập địa chỉ email, khung nhập mật khẩu và 1 nút để gửi yêu cầu đăng nhập.
- Đăng ký (Sign Up):
 - *Tính năng*: Server chỉ cho phép người dùng đăng ký duy nhất 1 địa chỉ email cho mỗi tài khoản, tên người dùng có thể giống nhau nhưng địa chỉ email không được giống nhau. Sau khi đăng ký thì có thể sử dụng ngay mà không cần phải đăng nhập lại. Nếu thông tin đăng nhập không hợp lệ, cảnh báo sẽ hiện lên. Ví dụ: mật khẩu phải có từ 7 ký tự trở lên.
 - *Giao diện*: Giao diện gồm khung nhập địa chỉ email (Email), tên người dùng (User Name), mật khẩu (Password), nhập lại mật khẩu (Confirm Password) và 1 nút để gửi yêu cầu đăng ký.
- Đăng xuất (Logout): Khi bấm đăng xuất, sẽ quay lại giao diện trang đăng nhập.
- Thao tác với văn bản: Nếu nhập quá ít ký tự (1 ký tự), cảnh báo sẽ hiện lên, hệ thống sẽ không ghi nhận nội dung vừa nhập. Database chỉ cho phép lưu trữ nội dung tối đa 10000 ký tự, nếu nhiều hơn thì vẫn có thể hiển thị trên bảng ghi nhưng không thể lưu lại trong database. Do lúc thiết kế cơ sở dữ liệu đã giới hạn số ký tự được phép ghi (bởi vì đây chỉ là ứng dụng ghi chú nên không cần cho phép ghi quá nhiều).

4.2. Sử dụng Docker cho ứng dụng

Cài đặt dockerfile:

- Dùng python 3.12.0 để code.

- Tạo 1 thư mục “app” để chạy trong container
- Copy thư mục instance sang thư mục “app”, trong thư mục instance có chứa file tdlst.db (file database).
- Dùng lệnh pip install -r requirements.txt để tự động import các thư viện đã trong file.
- Bắt đầu chạy từ file app.py để khởi tạo ứng dụng.

```

dockercfg docker-compose.yaml
1   version: '3.8'
2   services:
3     todolist:
4       build:
5         context: ./ 
6       ports:
7         - 8888:5000

```

Hình 39: Dockerfile

Cài đặt file docker-compose.yml

- Version cho docker engine là 3.8.
- Port để chạy trên máy là 5000, ánh xa ra bên ngoài sử dụng port 8000.

```

dockercfg dockerfile > ...
1   FROM python:3.12.0
2   WORKDIR /app
3   COPY . /app/
4   COPY instance /app/
5   RUN pip install -r requirements.txt
6   ENTRYPOINT [ "python" ]
7   CMD [ "app.py" ]

```

Hình 40: Docker-compose.yml

4.3. Hướng dẫn cài đặt

Tải các file về máy tính bằng đường link:

<https://drive.google.com/drive/folders/1sAa1Gh3aPaPxotOUbH7MBy15yd3GzzAu?usp=sharing>

Tiến hành build container (đây là app đơn giản nên chỉ sử dụng 1 container)

```
(myenv) PS D:\Todolist_Tutorial> docker-compose build --no-cache
2023/12/03 21:26:43 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 19.4s (10/10) FINISHED                                            docker:default
=> [todolist internal] load build definition from dockerfile                0.0s
=> => transferring dockerfile: 1868                                           0.0s
=> [todolist internal] load .dockerignore                                    0.0s
=> => transferring context: 28                                              0.0s
=> [todolist internal] load metadata for docker.io/library/python:3.12.0    1.0s
=> [todolist 1/5] FROM docker.io/library/python:3.12.0@sha256:1987c4ae3b5afaa3a7c5e247e9aab7348082ba167986c  0.0s
=> [todolist internal] load build context                                    0.1s
=> => transferring context: 203.39kB                                         0.1s
=> CACHED [todolist 2/5] WORKDIR /app                                       0.0s
=> [todolist 3/5] COPY . /app/                                             3.6s
=> [todolist 4/5] COPY instance /app                                       0.3s
=> [todolist 5/5] RUN pip install -r requirements.txt                      13.6s
```

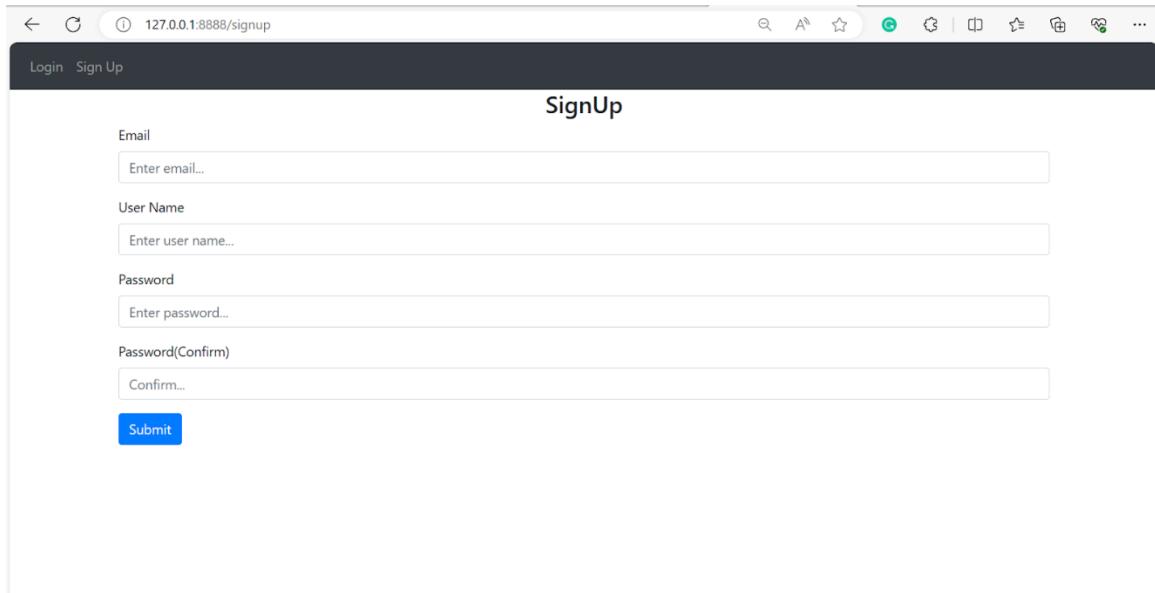
Hình 41: Build docker compose

Khởi động và chạy các dịch vụ đã được định nghĩa trong docker-compose.yml

```
(myenv) PS D:\Todolist_Tutorial> docker-compose up
[+] Building 0.0s (0/0)                                                       docker:default
[+] Running 1/1
✓ Container todolist_tutorial-todolist-1  Created                         0.1s
Attaching to todolist_tutorial-todolist-1
todolist_tutorial-todolist-1  | * Serving Flask app 'todolist'
todolist_tutorial-todolist-1  | * Debug mode: on
todolist_tutorial-todolist-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
todolist_tutorial-todolist-1  | * Running on all addresses (0.0.0.0)
```

Hình 42: Khởi động docker compose

Người dùng nhập văn bản vào ô lớn. Để lưu lại, nhấn nút “Add note”, nội dung vừa nhập sẽ hiện lên bảng ghi và lưu vào database của server. Nếu muốn xóa nội dung đã tồn tại trên bảng ghi, người dùng bấm vào nút “x” ở ngoài cùng bên phải của mỗi ô văn bản, nội dung đó cũng sẽ bị xóa khỏi bảng ghi và database.



Hình 43: Giao diện đăng ký

Hình 44: Giao diện đăng nhập

Item	Action
• dâx	X
• âsss	X
• dd	X
• ggggggggggg	X
• 12324	X
• sffsf	X

Hình 45: Giao diện chính

Các tài khoản đã đăng ký (dùng SQLite làm nơi lưu trữ tài khoản và nội dung ghi chú)
Khóa chính trong bảng User là id, được dùng để kết nối với bảng Note

user_id thể hiện người dùng nào đã nhập những nội dung nào.

user				
<input type="button" value="Reset Filters"/> <input type="button" value="Records: 1"/> Search 1 records...				
	id	email	password	user_name
	1	1 21522240@gm.uit...	pbkdf2:sha256:600...	khue

Hình 46: Database

	id	data	date	user_id
	1	dâx	2023-12-03 13:37:28	1
	2	âsss	2023-12-03 13:39:57	1
	3	dd	2023-12-03 14:17:23	1
	4	gggggggggg	2023-12-03 14:18:54	1

Hình 47: Các ghi chú được lưu trữ.

IV – Hướng dẫn sửa lỗi

1. Ứng dụng trò chơi “Revenge of the Rat”

1.1. Lỗi không thể sử dụng giao diện và âm thanh của máy chủ

Nếu chạy file main.py bị lỗi như hình bên dưới, hãy kiểm tra default device trên môi trường Ubuntu.

```
docker@ab6201530878:~$ python3 main.py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
ALSA lib confmisc.c:767:(parse_card) cannot find card '0'
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_card_driver returned error: No such file or directory
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_concat returned error: No such file or directory
ALSA lib confmisc.c:1246:(snd_func_refer) error evaluating name
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_refer returned error: No such file or directory
ALSA lib conf.c:5220:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM default
ALSA lib confmisc.c:767:(parse_card) cannot find card '0'
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_card_driver returned error: No such file or directory
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_concat returned error: No such file or directory
ALSA lib confmisc.c:1246:(snd_func_refer) error evaluating name
ALSA lib conf.c:4732:(_snd_config_evaluate) function snd_func_refer returned error: No such file or directory
ALSA lib conf.c:5220:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM default
Traceback (most recent call last):
  File "main.py", line 20, in <module>
    pygame.mixer.init()
pygame.error: No available audio device
```

Hình 48: Lỗi không thể sử dụng giao diện và âm thanh

Kiểm tra quyền truy cập của audio device bằng lệnh sau:

```
ls -l /dev/snd/
```

Nếu kết quả hiển thị như hình bên dưới có nghĩa là thiết bị của bạn chưa được cấp quyền truy cập cho “other” nên chúng không được phép chia sẻ với docker.

```
minhngoc@minhngoc-virtual-machine:~/Downloads$ ls -l /dev/snd
total 0
drwxr-xr-x 2 root root 60 Thg 12 1 22:12 by-path
crw-rw----+ 1 root audio 116, 6 Thg 12 1 22:12 controlC0
crw-rw----+ 1 root audio 116, 5 Thg 12 1 22:12 midiC0D0
crw-rw----+ 1 root audio 116, 3 Thg 12 1 22:13 pcmC0D0c
crw-rw----+ 1 root audio 116, 2 Thg 12 1 22:58 pcmC0D0p
crw-rw----+ 1 root audio 116, 4 Thg 12 1 22:13 pcmC0D1p
crw-rw----+ 1 root audio 116, 1 Thg 12 1 22:12 seq
crw-rw----+ 1 root audio 116, 33 Thg 12 1 22:12 timer
```

Hình 49: Kiểm tra quyền truy cập

Để cấp quyền truy cập vào các audio, hãy thêm lệnh các lệnh sau:

```
sudo chmod o+r /dev/snd/controlC0
sudo chmod o+r /dev/snd/midiC0D0
sudo chmod o+r /dev/snd/pcmC0D0c
sudo chmod o+r /dev/snd/pcmC0D0p
sudo chmod o+r /dev/snd/pcmC0D1p
sudo chmod o+r /dev/snd/seq
sudo chmod o+r /dev/snd/timer
```

Kiểm tra lại quyền truy cập bằng lệnh ls -l /dev/snd/: Nếu kết quả hiển thị như hình bên dưới là thành công và file main.py sẽ được thực thi như bình thường.

```
minhngoc@minhngoc-virtual-machine:~/Downloads$ ls -l /dev/snd
total 0
drwxr-xr-x 2 root root 60 Thg 12 1 22:12 by-path
crw-rw-rw++ 1 root audio 116, 6 Thg 12 1 22:12 controlC0
crw-rw-rw++ 1 root audio 116, 5 Thg 12 1 22:12 midiC0D0
crw-rw-rw++ 1 root audio 116, 3 Thg 12 1 23:16 pcmC0D0c
crw-rw-rw++ 1 root audio 116, 2 Thg 12 1 23:21 pcmC0D0p
crw-rw-rw++ 1 root audio 116, 4 Thg 12 1 23:16 pcmC0D1p
crw-rw-rw++ 1 root audio 116, 1 Thg 12 1 22:12 seq
crw-rw-rw++ 1 root audio 116, 33 Thg 12 1 22:12 timer
minhngoc@minhngoc-virtual-machine:~/Downloads$ █
```

Hình 50: Sửa lỗi không thể sử dụng giao diện và âm thanh

1.2. Lỗi không thể chạy file main.py

Nếu gặp lỗi như hình bên dưới, hãy cứ chạy lại lệnh python3 main.py một lần nữa, chương trình sẽ chạy bình thường

```

docker@7cc80e2d8ba5:~$ python3 -m main.py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: cHRM chunk does not match sRGB
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: cHRM chunk does not match sRGB
libpng warning: iCCP: known incorrect sRGB profile
X Error of failed request:  BadValue (integer parameter out of range for operation)
    Major opcode of failed request:  130 (MIT-SHM)
    Minor opcode of failed request:  3 (X_ShmPutImage)
    Value in failed request:  0x1f4
    Serial number of failed request:  13
    Current serial number in output stream:  14

```

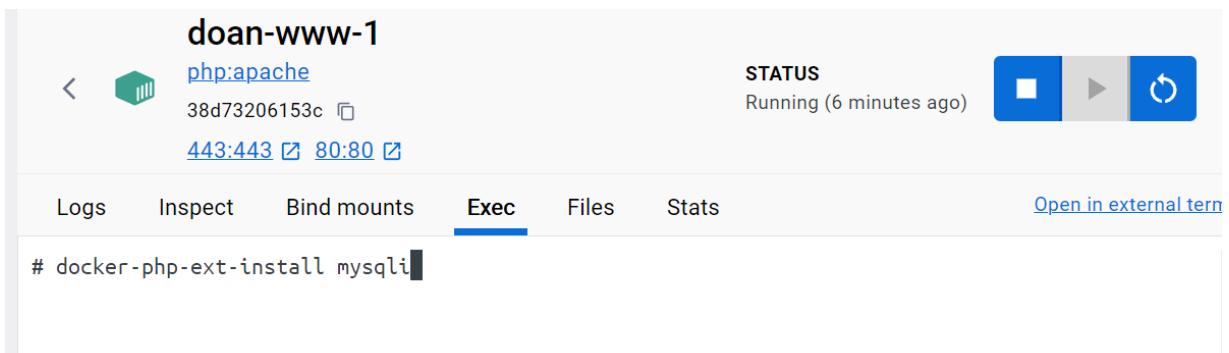
Hình 51: Lỗi không thể sử dụng trò chơi

2. Ứng dụng Blog cá nhân

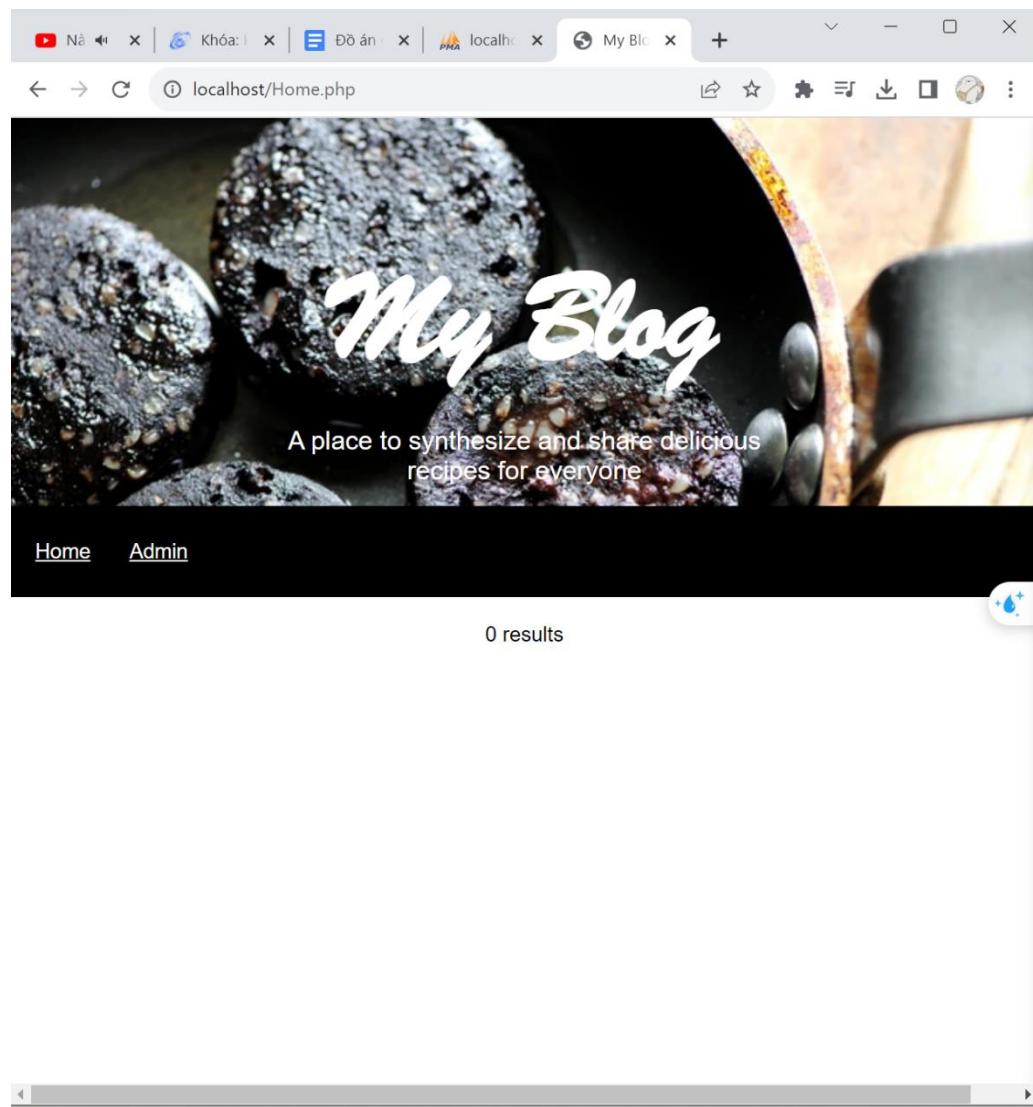
Truy cập vào localhost/Home.php để vào trang web của Blog. Nếu bị lỗi như hình bên dưới:

Fatal error: Uncaught Error: Call to undefined function mysqli_connect() in /var/www/html/Home.php:3 Stack trace: #0 {main} thrown in /var/www/html/Home.php on line 3

- Sửa lỗi: mở docker containers www -> chọn exec -> chạy lệnh docker-php-ext-install mysqli->restart lại containers



- Load lại trang



Phân công công việc

Sinh viên	MSSV	Công việc
Nguyễn Thị Hồng Lam	20521518	20%
Lê Đào Khánh Ngọc	20520653	10%
Mai Ngọc Phương Trinh	20520823	25%
Trần Lê Minh Ngọc	21521195	25%
Huỳnh Minh Khuê	21522240	20%

Tài liệu tham khảo

<https://github.com/riecho14/Docker-Dendam-Si-Tikus.git>

https://github.com/ThanhLa1802/web_framework/tree/master/FLASK_PROJECT/Todolist_Tutorial/todolist