

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN MÔN HỌC
MẠNG MÁY TÍNH
ĐỀ TÀI: Ứng dụng Socket – Địa điểm yêu thích

Giảng viên lý thuyết: Thầy Đỗ Hoàng Cường

Giảng viên hướng dẫn thực hành:

- Thầy Lê Hà Minh
- Thầy Nguyễn Thanh Quân

Lớp: 20TN

Thành viên thực hiện:

- 20120131 – Nguyễn Văn Lộc
- 20120536 – Võ Trọng Nghĩa
- 20120572 – Nguyễn Kiều Minh Tâm

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 4–5 NĂM 2022

Mục lục

1	Thông tin của nhóm	3
2	Mức độ hoàn thành	4
3	Kịch bản giao tiếp của chương trình	5
3.1	Giao thức trao đổi giữa client và server	5
3.2	Cấu trúc của thông điệp	5
3.3	Cách tổ chức cơ sở dữ liệu	6
3.4	Quá trình truyền từ client lên server	8
4	Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng	10
4.1	Hệ điều hành	10
4.2	Ngôn ngữ và các thư viện	10
5	Hướng dẫn sử dụng các tính năng của chương trình	11
5.1	Yêu cầu	11
5.2	Khởi động server	11
5.3	Khởi động client	11
5.4	Giao diện khởi động client	11
5.5	Truy vấn thông tin chi tiết một địa điểm	12
5.6	Tải tất cả hình ảnh đại diện	13
5.7	Tải các hình ảnh của một địa điểm	15
5.8	Demo	17
6	Phân công công việc	18

Danh sách hình vẽ

1	Ví dụ về cấu trúc lưu trữ dữ liệu	7
2	Ví dụ về cấu trúc thư mục con của <code>images</code>	8
3	Khởi động server	11
4	Khởi động client	11
5	Khởi động client	12
6	Truy vấn thông tin chi tiết một địa điểm	13
7	Tải tất cả hình ảnh đại diện	14
8	Tải tất cả hình ảnh đại diện	14
9	Tải tất cả hình ảnh đại diện	15
10	Tải các hình ảnh của một địa điểm	16
11	Tải các hình ảnh của một địa điểm	16
12	Tải các hình ảnh của một địa điểm	17

Danh sách bảng

1	Bảng thông tin thành viên	3
2	Bảng mức độ hoàn thành	4
3	Bảng phân công thành viên	18

1 Thông tin của nhóm

MSSV	Họ và tên	Email
20120131	Nguyễn Văn Lộc	20120131@student.hcmus.edu.vn
20120536	Võ Trọng Nghĩa	20120536@student.hcmus.edu.vn
20120572	Nguyễn Kiều Minh Tâm	20120572@student.hcmus.edu.vn

Bảng 1: Bảng thông tin thành viên

2 Mức độ hoàn thành

STT	Yêu cầu	Công việc	Đánh giá	Điểm tự đánh giá
1	Truy vấn danh sách địa điểm	Clients truy vấn được danh sách địa điểm do server quản lý. Các thông tin hiển thị: mã số địa điểm, tên địa điểm.	Hoàn thành	2
2	Truy vấn thông tin 1 địa điểm	Clients truy vấn được thông tin chi tiết về 1 địa điểm do server quản lý. Các thông tin hiển thị: mã số địa điểm, tên địa điểm, tọa độ địa điểm, mô tả.	Hoàn thành	2
3	Quản lý dữ liệu tại server bằng các loại file có cấu trúc như XML, JSON hoặc CSDL quan hệ	Dữ liệu được quản lý tại server bằng file JSON.	Hoàn thành	1
4	Cho phép tải hình ảnh đại diện từ server về client cho tất cả các địa điểm đang được quản lý.	Clients có thể tải về hình ảnh đại diện của các địa điểm từ server. Hình ảnh được hiển thị lên GUI của ứng dụng sau khi tải về.	Hoàn thành	2
5	Cho phép tải về các hình ảnh của 1 địa điểm từ server về client.	Clients có thể tải về các hình ảnh của một địa điểm từ server. Hình ảnh được hiển thị lên GUI của ứng dụng sau khi tải về.	Hoàn thành	2
6	Hỗ trợ nhiều clients truy cập đồng thời đến server.	Cho phép nhiều clients truy cập đồng thời đến server.	Hoàn thành	1

Bảng 2: Bảng mức độ hoàn thành

3 Kịch bản giao tiếp của chương trình

3.1 Giao thức trao đổi giữa client và server

Server và client trao đổi với nhau thông qua giao thức UDP.

3.2 Cấu trúc của thông điệp

Một thông điệp có tổng cộng 1024 bytes.

1. Đối với thông điệp từ server có 2 loại sau đây

- (a) *Thông điệp gửi số lượng thông điệp cần nhận cho data: |padding|len|*

Có dạng <1021 bytes 0>xyz (xyz là ba chữ số).

Gồm 2 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. padding: Gồm các null bytes (các bytes \x00) chèn vào trước trường len sao cho tổng thông điệp có đủ 1024 bytes.
- ii. len: kiểu str, thể hiện số lượng thông điệp (tối đa 3 bytes).

- (b) *Thông điệp gửi từng phần data: |ID|data|hash|*

Có dạng xyz<data: 981><hash: 40> (xyz là ba chữ số).

Gồm 3 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. ID: Kiểu str gồm 3 ký tự thể hiện số trong đoạn [000; 999] (3 bytes).
- ii. data: Kiểu byte, gồm 981 (1024 - 43) ký tự (981 bytes).
- iii. hash: Kiểu str, gồm 40 ký tự (40 bytes) được tạo khi hash data bằng thuật toán SHA1 để kiểm lỗi. 40 bytes này mô tả một số nguyên 20 bytes = 160 bits trong hệ thập lục phân.

2. Đối với thông điệp từ client có 6 loại sau đây.

- (a) *Thông điệp xác nhận số lượng: |specifier|len|padding|*

Có dạng ACK_LEN_xyz<1013 bytes 0> (xyz là ba chữ số).

Gồm 3 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. specifier: kiểu str, là chuỗi ký tự "ACK_LEN_" (8 bytes)
- ii. len: kiểu str, là số lượng thông điệp đã nhận từ server, gồm 3 ký tự thể hiện số trong đoạn [000; 999] (3 bytes)
- iii. padding: gồm 1013 (1024 - 8 - 3) bytes \x00.

- (b) *Thông điệp xác nhận phần data: |specifier|id|padding|*

Có dạng ACK_xyz<1017 bytes 0> (xyz là ba chữ số).

Gồm 3 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. specifier: kiểu str, là chuỗi ký tự "ACK_" (4 bytes)
- ii. id: kiểu str, là thứ tự của thông điệp nhận từ server, gồm 3 ký tự thể hiện số trong đoạn [000; 999] (3 bytes)
- iii. padding: gồm 1017 (1024 - 4 - 3) bytes \x00.

(c) *Thông điệp yêu cầu thông tin toàn bộ địa điểm: |command|padding|*

Có dạng `GIV_ALL<1017 bytes 0>`

Gồm 2 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. **command**: kiểu **str**, là chuỗi ký tự "GIV_ALL" (7 bytes)
- ii. **padding**: gồm 1017 (1024 - 7) bytes `\x00`.

(d) *Thông điệp yêu cầu thông tin chi tiết 1 địa điểm: |command|id|padding|*

Có dạng `GIV_DETAIL_<id: x bytes><1013 - x bytes 0>`

Gồm 3 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. **command**: kiểu **str**, là chuỗi ký tự "GIV_DETAIL_" (11 bytes)
- ii. **id**: kiểu **str**, thể hiện id của địa điểm cần lấy.
- iii. **padding**: Gồm các bytes `\x00` điền cho đủ 1024 bytes.

(e) *Thông điệp yêu cầu ảnh đại diện 1 địa điểm: |command|id|padding |*

Có dạng `GIV_AVT_<id: x bytes><1016 - x bytes 0>`

Gồm 3 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. **command**: kiểu **str**, là chuỗi ký tự "GIV_AVT_" (8 bytes)
- ii. **id**: kiểu **str**, thể hiện id của địa điểm cần lấy.
- iii. **padding**: Gồm các bytes `\x00` điền cho đủ 1024 bytes.

(f) *Thông điệp yêu cầu ảnh tại 1 địa điểm: |command|position|separator|id|padding|*

Có dạng `GIV_IMG_xyz_<id: t bytes><1012 - t bytes 0>` (xyz là ba chữ số).

Gồm 5 trường theo thứ tự đó với các mô tả chi tiết sau đây:

- i. **command**: kiểu **str**, là chuỗi ký tự "GIV_IMG_" (8 bytes)
- ii. **position**: kiểu **str**, là thứ tự trong danh sách ảnh cần lấy, gồm 3 ký tự thể hiện số trong đoạn [000; 999] (3 bytes)
- iii. **separator**: kiểu **char**, là ký tự '_'
- iv. **id**: kiểu **str**, thể hiện id của địa điểm cần lấy.
- v. **padding**: Gồm các bytes `\x00` điền cho đủ 1024 bytes.

3.3 Cách tổ chức cơ sở dữ liệu

Dữ liệu được lưu trong tập tin **db.json** thành các bộ với các trường thông tin sau:

- **id**: gồm ba ký tự kiểu **char**, là viết tắt tên địa điểm.
- **name**: tên đầy đủ của địa điểm tương ứng.
- **coordinate**: tọa độ vị trí trung tâm của địa điểm theo thứ tự vĩ độ và kinh độ. Địa điểm có vĩ độ dương nằm ở phía Bắc đường xích đạo, vĩ độ âm nằm ở phía Nam đường xích đạo. Địa điểm có kinh độ dương nằm ở phía Đông kinh tuyến gốc, kinh độ âm nằm ở phía Tây kinh tuyến gốc.

- **description**: đoạn mô tả ngắn về địa điểm tương ứng.
- **avatar**: lưu đường dẫn tương đối (so với tập tin **db.json**) đến ảnh đại diện của địa điểm.
- **images**: lưu danh sách đường dẫn tương đối (so với tập tin **db.json**) đến các hình ảnh của địa điểm.

Thư mục **images** được chia thành các thư mục con, mỗi thư mục con có tên là các **id**, trong thư mục tương ứng bao gồm một hình đại diện **avt.jpg**, và các hình ảnh minh họa khác **img*.jpg**, trong đó * là số thứ tự của hình ảnh tương ứng. Các thông tin được đưa vào cơ sở dữ liệu dưới dạng mã Unicode.

```
{
  "id": "TRV",
  "name": "Tr\u00e0 Vinh",
  "coordinate": [
    9.94719,
    106.34225
  ],
  "description": "Tr\u00e0 Vinh l\u00e0 m\u1ed9t t\u1ec9nh ven bi\u1ec3n thu\u1ed9c v\u00f9ng",
  "avatar": "./images/TRV/avt.jpg",
  "images": [
    "./images/TRV/img1.jpg",
    "./images/TRV/img2.jpg",
    "./images/TRV/img3.jpg",
    "./images/TRV/img4.jpg",
    "./images/TRV/img5.jpg",
    "./images/TRV/img6.jpg",
    "./images/TRV/img7.jpg",
    "./images/TRV/img8.jpg",
    "./images/TRV/img9.jpg"
  ]
},
```

Hình 1: Ví dụ về cấu trúc lưu trữ dữ liệu

This PC > HDD (F:) > GitHub > HCMUS-Computer-Networks-Projects > Socket-Place > server > images > TRV

Name	Date	Type	Size	Tags
avt.jpg	13/05/2022 10:36 AM	JPG File	207 KB	
img1.jpg	20/05/2022 4:59 PM	JPG File	449 KB	
img2.jpg	13/05/2022 10:36 AM	JPG File	247 KB	
img3.jpg	13/05/2022 10:36 AM	JPG File	74 KB	
img4.jpg	13/05/2022 10:36 AM	JPG File	60 KB	
img5.jpg	13/05/2022 10:37 AM	JPG File	866 KB	
img6.jpg	13/05/2022 10:37 AM	JPG File	129 KB	
img7.jpg	13/05/2022 10:37 AM	JPG File	118 KB	
img8.jpg	01/11/2002 10:35 AM	JPG File	292 KB	
img9.jpg	13/05/2022 10:38 AM	JPG File	304 KB	

Hình 2: Ví dụ về cấu trúc thư mục con của images

3.4 Quá trình truyền từ client lên server

1. Đoạn dữ liệu ban đầu sẽ được xử lý thành danh sách các thông điệp kích thước 1024.
2. Server gửi số lượng thông điệp cho client và chờ **ACK**.
3. Sử dụng thuật toán **Sliding window** với kích thước 5 để truyền:

- Phía server

Algorithm 1 Phía server

```

1: function SWSERVER
2:    $tds \leftarrow$  danh sách thông điệp cần truyền
3:    $l\_tds \leftarrow$  độ dài danh sách
4:    $size \leftarrow$  kích thước cửa sổ = 5
5:    $i \leftarrow 0$ 
6:   while  $i < l\_tds$  do
7:      $struyen \leftarrow$  số lượng truyền =  $\min(size, l\_tds - i)$ 
8:      $kt \leftarrow$  mảng đánh dấu kích thước  $struyen$ 
9:     while  $kt$  chưa được đánh dấu hết do
10:      truyền lần lượt toàn bộ cửa sổ hiện tại ( $i \rightarrow i + struyen$ )
11:      thực hiện nhận ACK  $struyen$  lần, đánh dấu  $ID$  vào  $kt$ 
12:      nếu quá giờ nhận ACK, coi như gói bị mất và truyền lại
13:       $i \leftarrow i + size$  (dịch tới cửa sổ tiếp theo)
```

- Phía client

Algorithm 2 Phía client

```
1: function SWCLIENT
2:    $l\_tds \leftarrow$  độ dài danh sách
3:    $size \leftarrow$  kích thước cửa sổ = 5
4:    $i \leftarrow 0$ 
5:    $result \leftarrow$  thông điệp được truyền
6:   while  $i < l\_tds$  do
7:      $snhan \leftarrow$  số lượng nhận =  $\min(size, l\_tds - i)$ 
8:      $buffer \leftarrow$  bộ nhớ tạm để lưu cửa sổ với kích thước  $snhan$ 
9:     while  $buffer$  chưa đầy hết: do
10:      với mỗi phần tử, thực hiện nhận và kiểm lỗi, sau đó đưa
      vào  $buffer$ 
11:      sắp xếp lại gói tin trong  $buffer$ , tách phần  $data$  và nối vào
       $result$ 
12:       $i \leftarrow i + size$  (dịch tới cửa sổ tiếp theo)
```

4 Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng

4.1 Hệ điều hành

Ứng dụng được lập trình trên hệ điều hành Windows 11.

4.2 Ngôn ngữ và các thư viện

Ứng dụng được lập trình bằng ngôn ngữ Python 3.

Các thư viện sử dụng trong chương trình:

- Thư viện **json** dùng để giao tiếp với cơ sở dữ liệu.
- Thư viện **socket** cho việc trao đổi dữ liệu giữa client và server.
- Thư viện **tkinter**, **PIL** cho việc xây dựng giao diện.
- Các thư viện khác: **queue**, **logging**, **random**, **string**, **tempfile**, **hashlib**, **ctypes**, **functools**.

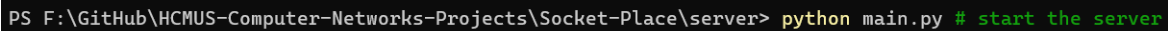
5 Hướng dẫn sử dụng các tính năng của chương trình

5.1 Yêu cầu

Để sử dụng được chương trình, máy tính phải cài đặt Python 3 và các thư viện ở phần 4.

5.2 Khởi động server

Để khởi động server, ta vào đường dẫn chứa các tập tin mã nguồn của server, mở terminal và sử dụng câu lệnh: `python main.py` hoặc `python3 main.py`.

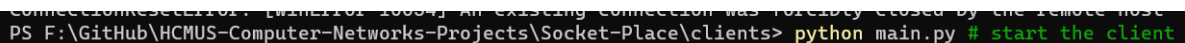


```
PS F:\GitHub\HCMUS-Computer-Networks-Projects\Socket-Place\server> python main.py # start the server
```

Hình 3: Khởi động server

5.3 Khởi động client

Để khởi động client, ta vào đường dẫn chứa các tập tin mã nguồn của client, mở terminal và sử dụng câu lệnh: `python main.py` hoặc `python3 main.py`.

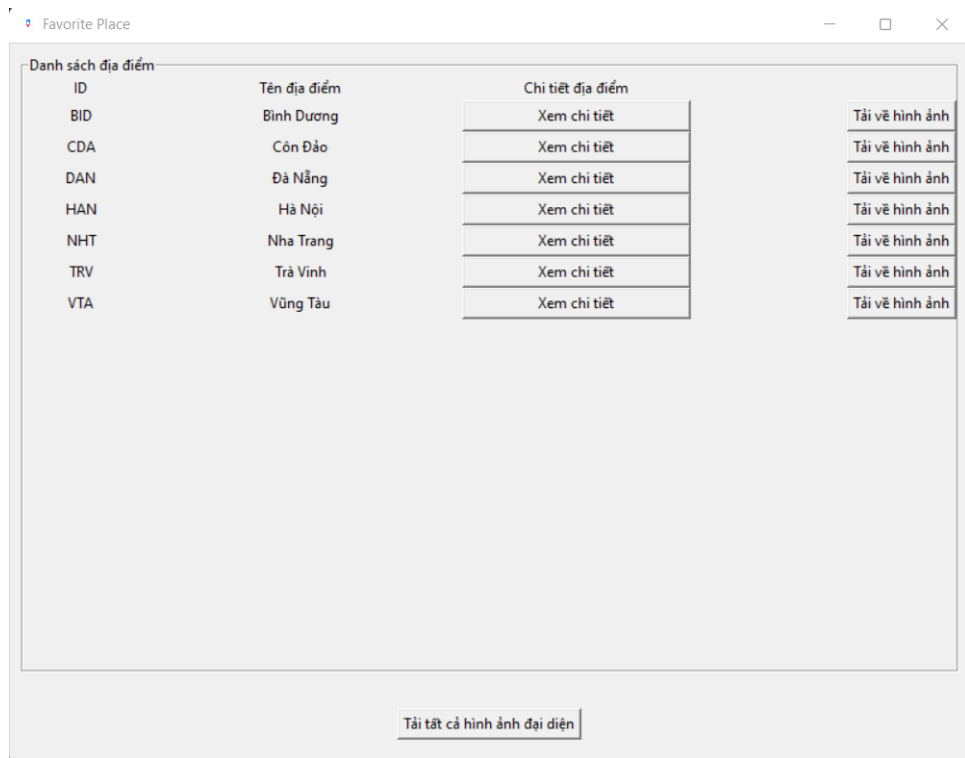


```
ConnectionResetError: [WinError 10054] An existing connection was forcibly closed by the remote host
PS F:\GitHub\HCMUS-Computer-Networks-Projects\Socket-Place\clients> python main.py # start the client
```

Hình 4: Khởi động client

5.4 Giao diện khởi động client

Sau khi khởi động, client có giao diện như sau:



Hình 5: Khởi động client

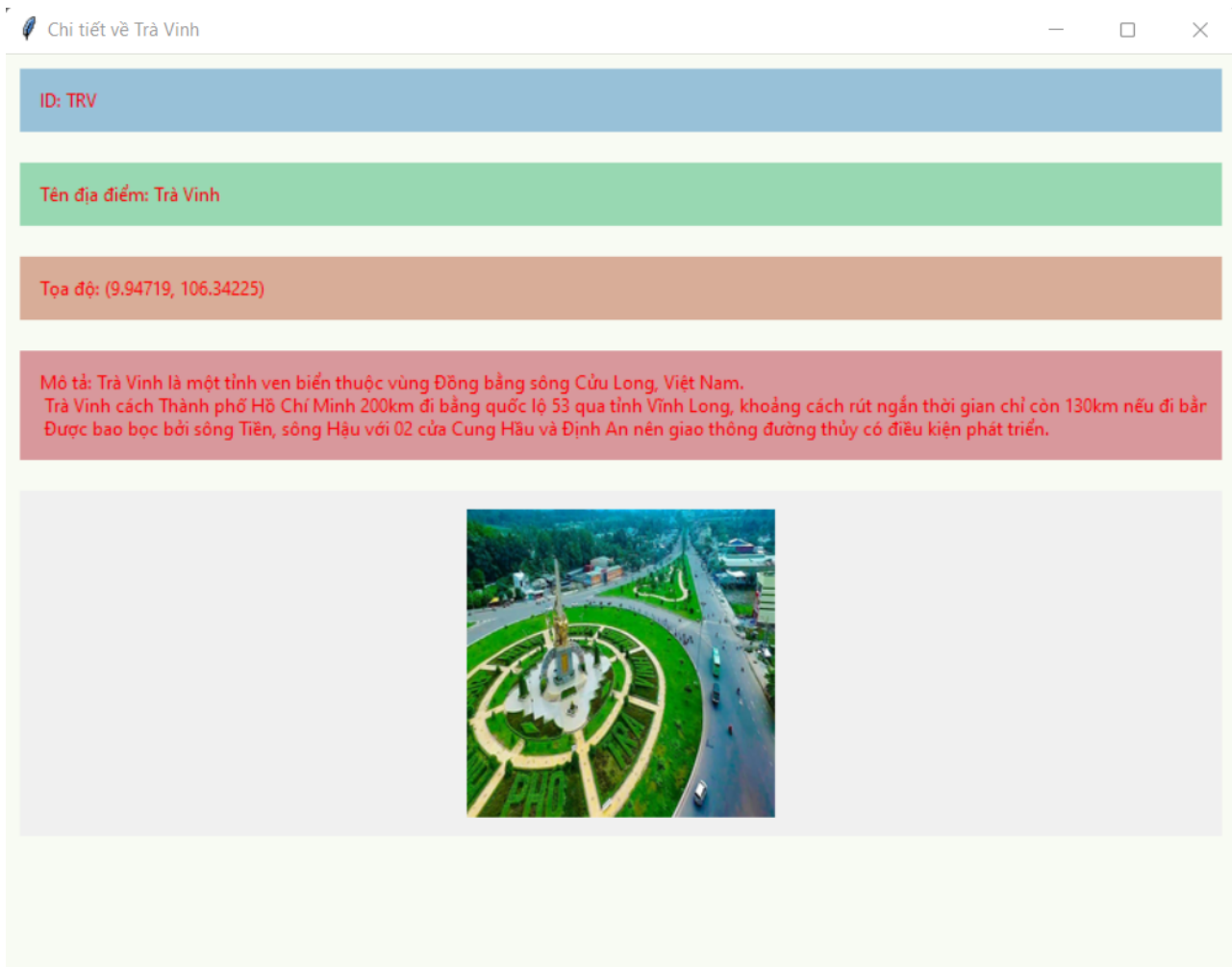
Giao diện khởi đầu của client là danh sách các địa điểm đang được server quản lý, các thông tin hiển thị bao gồm: ID địa điểm và tên địa điểm.

Trên giao diện khởi đầu, nút **Xem chi tiết** bên cạnh tên mỗi địa điểm cho phép ta truy vấn thông tin chi tiết về địa điểm đó từ server; nút **Tải về hình ảnh** bên cạnh mỗi địa điểm cho phép ta tải về các hình ảnh của địa điểm đó từ server, hình ảnh sau khi tải về được lưu trong thư mục **Temp** và được hiển thị lên GUI.

Ở phía dưới cùng của giao diện là nút **Tải tất cả hình ảnh đại diện**, cho phép ta tải về hình ảnh đại diện của tất cả các địa điểm đang được quản lý bởi server. Hình ảnh sau khi tải về được xử lý tương tự bên trên.

5.5 Truy vấn thông tin chi tiết một địa điểm

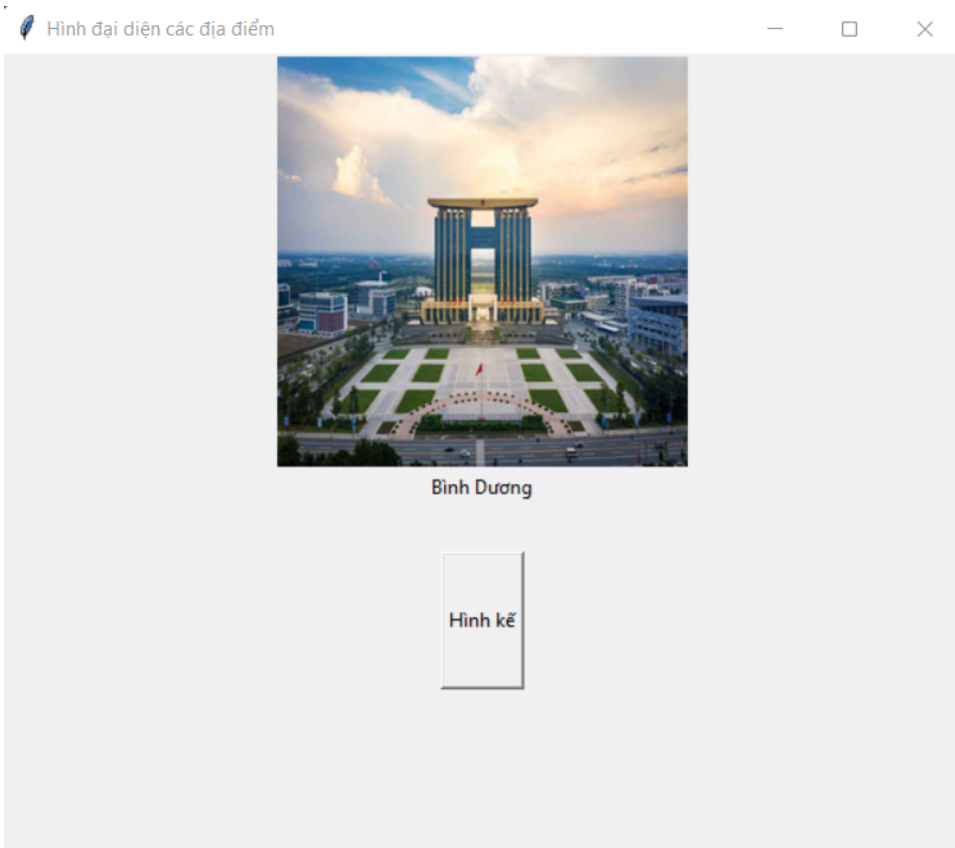
Nút **Xem chi tiết** ở giao diện khởi đầu cho phép client truy vấn thông tin chi tiết, được lưu tại server, về địa điểm tương ứng. Các thông tin hiển thị lên màn hình gồm: mã số địa điểm, tên địa điểm, tọa độ (vĩ độ, kinh độ) của địa điểm, mô tả và hình đại diện. Hình 6 là ví dụ về truy vấn thông tin chi tiết một địa điểm.



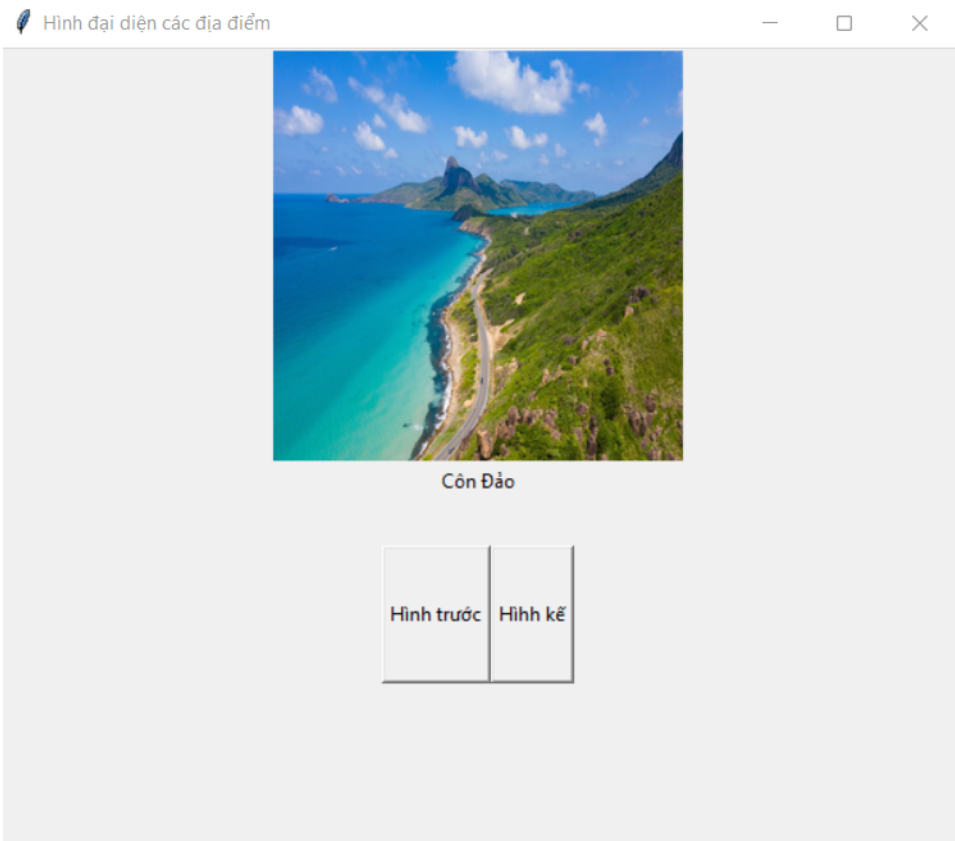
Hình 6: Truy vấn thông tin chi tiết một địa điểm

5.6 Tải tất cả hình ảnh đại diện

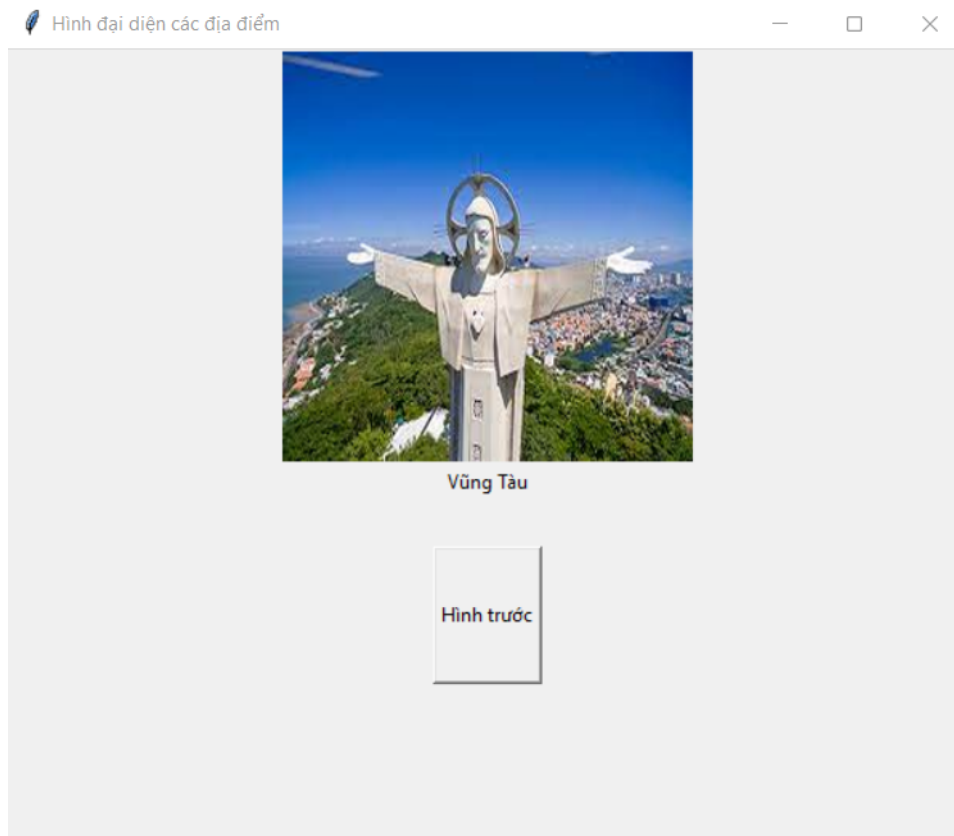
Nút **Tải tất cả hình ảnh đại diện** cho phép client tải về tất cả hình ảnh đại diện của các địa điểm từ server. Hình ảnh sau khi tải về được lưu trong thư mục **Temp** và được hiển thị lên giao diện của client. Các nút **Hình kế** và **Hình trước** tương ứng cho ta xem hình ảnh kế tiếp và hình ảnh ngay trước của hình ảnh hiện tại. Hình 7, 8 và 9 là ví dụ về tải tất cả hình ảnh đại diện.



Hình 7: Tải tất cả hình ảnh đại diện



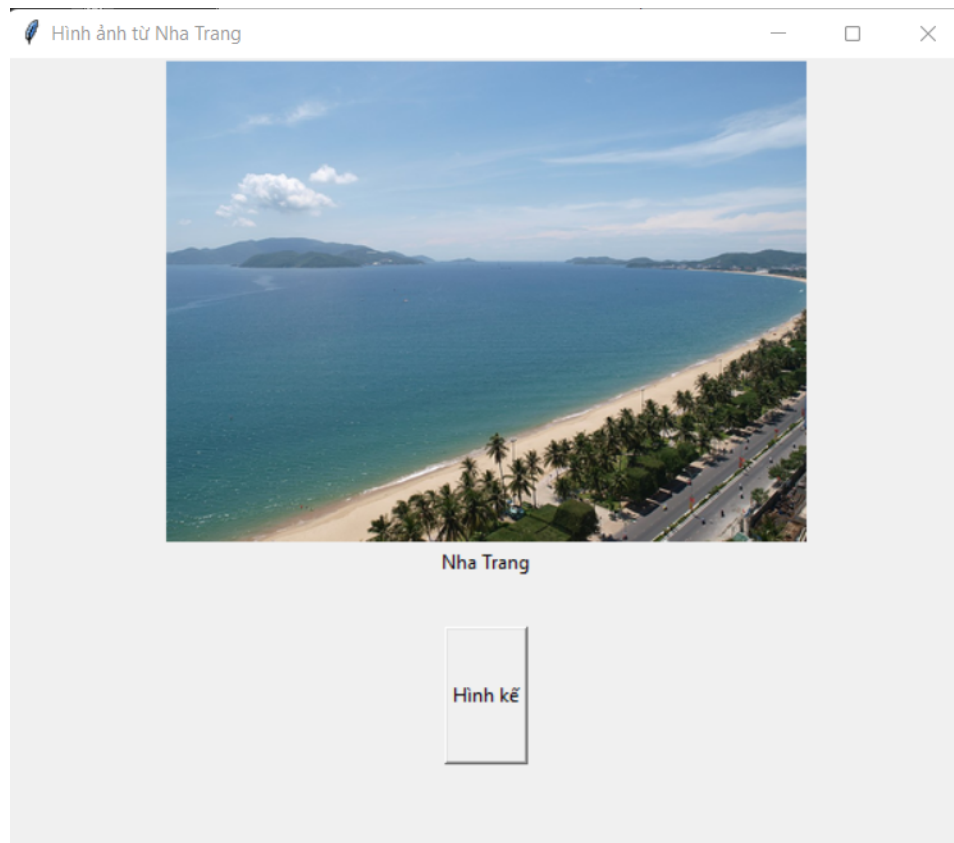
Hình 8: Tải tất cả hình ảnh đại diện



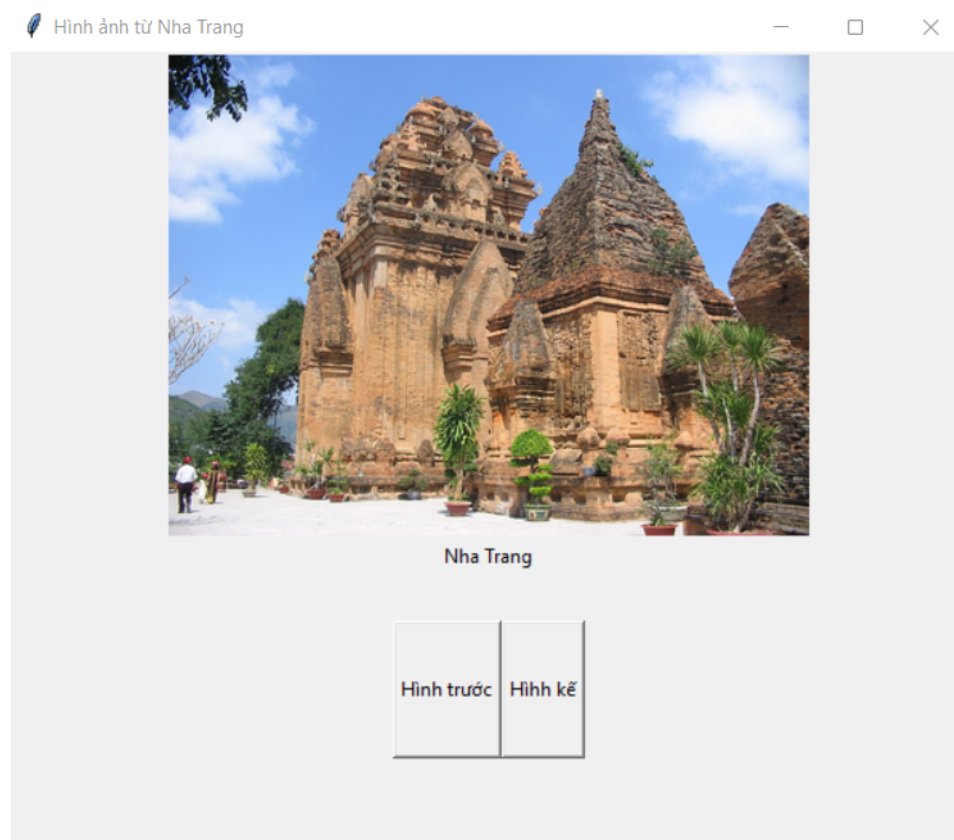
Hình 9: Tải tất cả hình ảnh đại diện

5.7 Tải các hình ảnh của một địa điểm

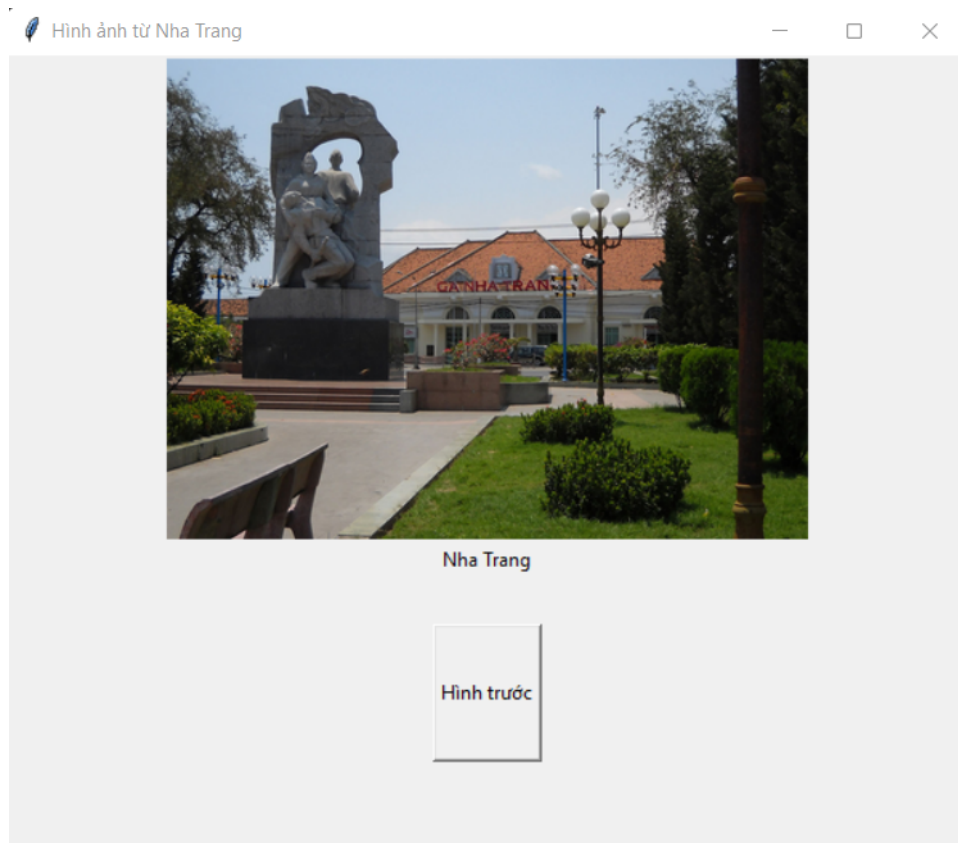
Nút **Tải về hình ảnh** bên cạnh mỗi địa điểm cho phép client tải về tất cả hình ảnh của một địa điểm từ server. Hình ảnh sau khi tải về được xử lý tương tự như phần 5.6. Các nút trong giao diện hiển thị hình ảnh cũng tương tự. Hình 10, 11 và 12 là ví dụ về tải về các hình ảnh của một địa điểm.



Hình 10: Tải các hình ảnh của một địa điểm



Hình 11: Tải các hình ảnh của một địa điểm



Hình 12: Tải các hình ảnh của một địa điểm

5.8 Demo

Demo của chương trình có thể được xem tại [đây](#).

Link: <https://www.youtube.com/watch?v=K0q0q73HTxc>

6 Phân công công việc

MSSV	Họ và tên	Công việc	Phần trăm đóng góp
20120131	Nguyễn Văn Lộc	Thiết kế các hàm ở client và GUI	35
20120536	Võ Trọng Nghĩa	Thiết kế giao thức truyền dữ liệu giữa client và server	45
20120572	Nguyễn Kiều Minh Tâm	Thiết kế GUI	20

Bảng 3: Bảng phân công thành viên

Tài liệu tham khảo chủ yếu

- [1] ActiveState. *How To Display Data In A Table Using Tkinter*. URL: <https://www.activestate.com/resources/quick-reads/how-to-display-data-in-a-table-using-tkinter/>.
- [2] David Amos. *Python GUI Programming With Tkinter*. URL: <https://realpython.com/python-gui-tkinter/#getting-multiline-user-input-with-text-widgets>.
- [3] Coder Arts. *How to implement Keyboard and Mouse click event in tkinter*. URL: <https://www.codersarts.com/post/2019/05/30/how-to-implement-keyboard-and-mouse-click-event-in-tkinter-tkinter-event-handling-event-a>.
- [4] DelftStack. *Pass Arguments to Tkinter Button Command*. URL: <https://www.delftstack.com/howto/python-tkinter/how-to-pass-arguments-to-tkinter-button-command/>.
- [5] Bijay Kumar. *Python Tkinter Image + Examples*. URL: <https://pythonguides.com/python-tkinter-image/>.
- [6] Bijay Kumar. *Python Tkinter Table Tutorial*. URL: <https://pythonguides.com/python-tkinter-table-tutorial/>.