ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Mạng Máy Tính - Computer Network - 191

Assignment 1

Socket Chat Application

GVHD: PhD. Phạm Trần Vũ
PhD. Nguyễn Mạnh Thìn

Sinh Viên:

Hoàng Vũ Trọng Thụy - 1710321
Lê Trung Vinh - 1710388

Mục lục

Danh sách hình vẽ

1 Định nghĩa chức năng

1.1 Đăng nhập/đăng xuất

- Cho phép user nhập tên và nhập đường dẫn tới server và tên port để vào phòng chat. Tên user phải bắt đầu bằng chữ cái
- Tạo avatar là chữ cái đầu tên của user

1.2 Gửi tin nhắn công khai trong phòng chat chung

- Cho phép user gửi tin nhắn đến cùng lúc các user khác thông qua một kênh chat chung.
- Tin nhắn gửi đi có thể là tin nhắn văn bản hoặc tin nhắn thoại.

1.3 Gửi file

- Cho phép user gửi một tệp tin từ trong máy tính cá nhân tới các user khác trong phòng chat chung hoặc tới một user khác trong server.

1.4 Xem trạng thái online offline của bạn bè trong list danh sách

- User có thể thấy ai đang online nếu người đó có tích xanh trước tên của mình, offline nếu tích xám ở bảng danh sách user và away nếu tích có màu vàng.
- User cũng có thể đổi trạng thái sử dụng của mình về online, away, busy.

1.5 Trao đổi riêng tư giữa hai người dùng (cần server xử lý)

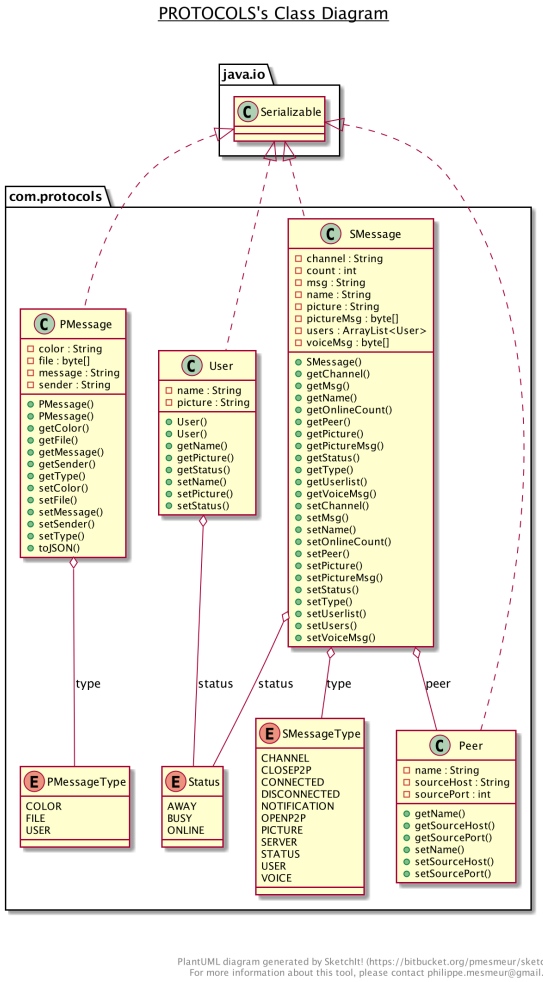
- Cho phép 2 user trong phòng chat nhắn tin qua lại với nhau.
- Tin nhắn gửi đi có thể là tin nhắn văn bản hoặc tin nhắn thoại.

1.6 Trao đổi riêng tư trực tiếp giữa hai người dùng (không thông qua server)

- Cho phép 2 user thiết lập một kênh giao tiếp riêng mà không cần đến server hỗ trợ.
- Tin nhắn gửi đi có thể là tin nhắn văn bản hoặc tin nhắn thoại.
- User có thể đổi màu tin nhắn

2 Định nghĩa giao thức

Ứng dụng sử dụng 2 giao thức: giao thức sử dụng trong giao tiếp giữa client-server (dựa trên giao thức TCP) và giao thức sử dụng trong giao tiếp giữa client-client (dựa trên giao thức UDP).



Hình 1: Mô hình định nghĩa giao thức giao tiếp trong ứng dụng

2.1 Giao thức giao tiếp client-server

Client giao tiếp với Server để thức hiện các chức năng: thông báo đăng nhập/đăng xuất; thay đổi trạng thái hoạt động; gửi công khai/riêng tư tin nhắn, ảnh, âm thanh sử dụng giao thức TCP; thiết lập kết nối UDP trực tiếp với client khác.

Giao thức gồm các trường thông tin chính sau:

1. **name: String.** Thông tin về client (tên đăng nhập).
2. **type: SMessageType.** Chức năng yêu cầu server xử lý, gồm có:
   - DISCONNECTED
   - CONNECTED
   - STATUS
   - USER
   - SERVER
   - NOTIFICATION

- VOICE
- CHANNEL
- PICTURE.
- OPENP2P
- CLOSEP2P

3. **channel: String.** Thông tin về **kênh** đang giao tiếp, gồm: **#Community** - tin nhắn gửi công khai đến toàn bộ người dùng, **Personal** - tin nhắn gửi riêng đến người dùng xác định (toàn bộ dựa trên giao thức TCP, tức là client - server - client).
4. **count: Integer.** Số lượng người dùng.
5. **status: Status.** Trạng thái hoạt động của người dùng: ONLINE, AWAY, BUSY.
6. **users: User.** Danh sách người dùng.
7. **picture: String.** Đường dẫn đến ảnh người dùng.
8. **msg: String.** Nội dung tin nhắn văn bản.
9. **pictureMsg: byte[].** Nội dung của ảnh được mã hoá theo chuẩn Base64.
10. **voiceMsg: byte[].** Nội dung của âm thanh được mã hoá.

2.2 Giao thức giao tiếp client-client

Giao thức sử dụng trong giao tiếp trực tiếp giữa 2 client với nhau (dựa trên UDP):

1. **sender: String.** Thông tin người gửi (tên).
2. **message: String.** Nội dung tin nhắn văn bản.
3. **type: PMessageType.** Chức năng yêu cầu, gồm: USER, FILE, COLOR.
4. **file: byte[].** Nội dung file được mã hoá theo chuẩn Base64.
5. **color: String.** Mã hex của màu cuộc trò chuyện.

3 Miêu tả chi tiết hiện thực chức năng

3.1 Chức năng trên client-server

Server có thể phục vụ cùng lúc nhiều yêu cầu từ client, tạo nhiều thread, trong đó, 1 thread **master** chuyên ghi nhận các socket kết nối đến server và nhiều thread (các **handle**) xử lý các yêu cầu chức năng tương ứng với từng client trong quá trình giao tiếp.

Các chức năng yêu cầu được xác định dựa vào trường thông tin **type.** của mỗi message gửi từ client. Message sẽ được truyền theo đường đi client-server-clients.

3.1.1 Đăng nhập/đăng xuất ứng dụng.

Server cho phép người dùng sử dụng một tên đăng nhập để đăng ký vào Lobby của ứng dụng. Tên đăng nhập này là duy nhất.

Server sẽ lưu trữ các người dùng thông qua HashMap từ `username` đến user tương ứng. Nhờ đó trên mỗi **handle** chỉ cần lưu một trường thông tin là `username` vẫn có thể truy xuất thông tin về người dùng cũng như kiểm tra tên đăng nhập đã tồn tại hay chưa.

Khi người dùng đăng nhập/đăng xuất, một message có **kiểu** là *CONNECTED/DISCONNECTED* sẽ được gửi từ client đến server, server sẽ ghi nhận và thông báo đến toàn bộ #Community (cùng kiểu với message nhận từ client) để cập nhận danh sách trạng thái trên khung hiện thị của mỗi client.

3.1.2 Thay đổi trạng thái hoạt động

Thông tin trạng thái hoạt động của người dùng được lưu tại trường thông tin **status.** trên mỗi message mỗi khi client gửi yêu cầu thay đổi trạng thái đến server.

Server sẽ ghi nhận và gửi đến toàn # Community (kể cả client gửi request) để cập nhật trạng thái hoạt động hiện thị trên khung lobby của mỗi client.

3.1.3 Gửi công khai tin nhắn văn bản/ảnh/âm thanh

Nội dung tin nhắn văn bản hoặc ảnh hoặc âm thanh được mã hoá và lưu trữ tại các trường **msg.** hoặc **pictureMsg.** hoặc **voiceMsg.**.

Server ghi nhận và gửi response đến toàn bộ #Community.

3.1.4 Gửi riêng tư tin nhắn văn bản hoặc ảnh hoặc âm thanh.

Người dùng có thể tuỳ ý thay đổi kênh giao tiếp bằng cách trỏ đến các kênh xác định có trên khung chat. Một message mang theo thông tin kênh lưu trữ tại trường **channel.** được gửi đến Server (một **handle**). Trên server cũng sẽ có một trường lưu trữ thông tin về kênh giao tiếp hiện tại đang phục vụ.

Việc cập nhật kênh giao tiếp sẽ được thức hiện trước khi việc gửi tin nhắn riêng tư được thức hiện. Điểm khác biệt khi gửi tin nhắn riêng tư (so với gửi công khai) là chỉ người nhận (và người gửi) nhận được tin nhắn. Đường đi của message sẽ đi từ người gửi - server - (người nhận - người gửi).

3.1.5 Yêu cầu mở/tắt giao tiếp UDP giữa 2 client

Khi người dùng muốn thiết lập một giao tiếp UDP với người dùng khác. Một message có **type.** là **SMessageType.OPENP2P** kèm theo thông tin về người dùng đầu bên kia (tên đăng nhập) được lưu trữ tại trường **peer.** được gửi theo đường *client 1 - server - client 2*. Client 2 nhận message sẽ gửi lại một message tương ứng đến client 1. và giao tiếp UDP được thiết lập. Trường này lưu trữ ba thông tin:

- **name.** Tên đăng nhập của người dùng bên kia.
- **host.** Địa chỉ IP của người dùng bên này.
- **port.** Port gửi của người dùng bên này.

3.2 Các chức năng trên client-client

Khác với mô hình client-server, mô hình client-client sử dụng giao thức UDP. Trên mỗi client sẽ có hai thanh phần, một cho xử lý thông tin nhận vào **Receiver**, một xử lý thông tin gửi đi **Sender**.

Tuỳ vào từng chức năng mà người dùng thực hiện, **sender** sẽ đóng gói thông tin trong protocol-message và gửi đến trực tiếp người dùng đang giao tiếp bên đầu kia. **Receiver** nhận gói tin và trích các thông tin trên đó để ghi nhận và xử lý phù hợp. Để rõ hơn ta sẽ đi vào từng chức năng cụ thể:

3.2.1 Gửi tin nhắn văn bản

Sender sẽ đóng gọi protocol-message gồm các 5 trường thông tin (như mô tả ở trên):

- **sender.** lưu thông tin về người gửi.
- **type.** gắn thông tin PMessageType.USER - giao tiếp bằng tin nhắn văn bản.
- **message.** gắn nội dụng tin nhắn.
- **file.** và **color.** được để trống (null).

Receiver sẽ nhận protocol-message sau đó ghi nhận chức năng bằng trường thông tin **type.**

3.2.2 Gửi file

Sender sẽ mã hoá nội dung file theo chuẩn Base64 và lưu trữ nó trong trường thông tin **file.** của protocol-message.

Kèm theo tên file được lưu vào trường thông tin **message.** Lúc này **type.** sẽ giữ giá trị `PMessageType.FILE`.

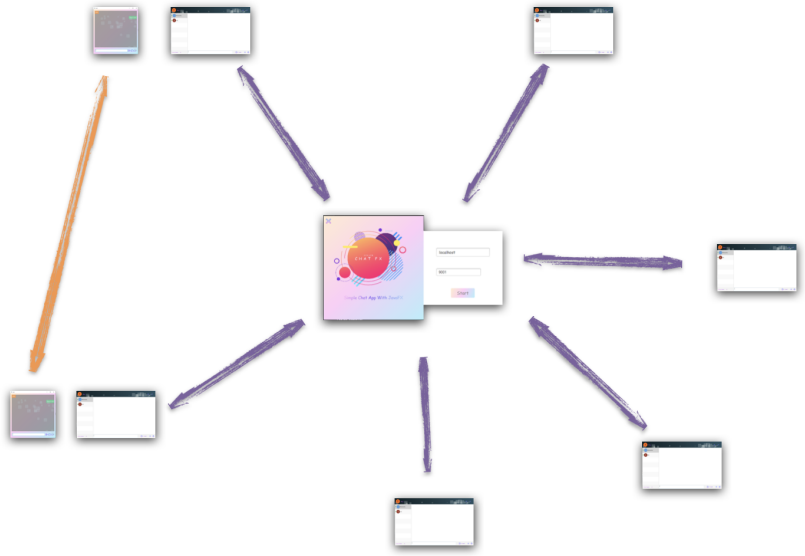
Receiver nhận message từ **sender** bên gửi, ghi nhận thông tin tại trường **type.** để có cách phù hợp xử lý gói tin đến.

3.2.3 Đổi màu cuộc trò chuyện

Người dùng sẽ chọn màu hiển thị khi mở bảng màu trên GUI, **sender** ghi nhận màu dưới dạng HEX và lưu thông tin tại trường **color.** Lúc này, **type.** sẽ giữ giá trị `PMessageType.COLOR`.

4 Thiết kế chi tiết

4.1 Kiến trúc tổng quát



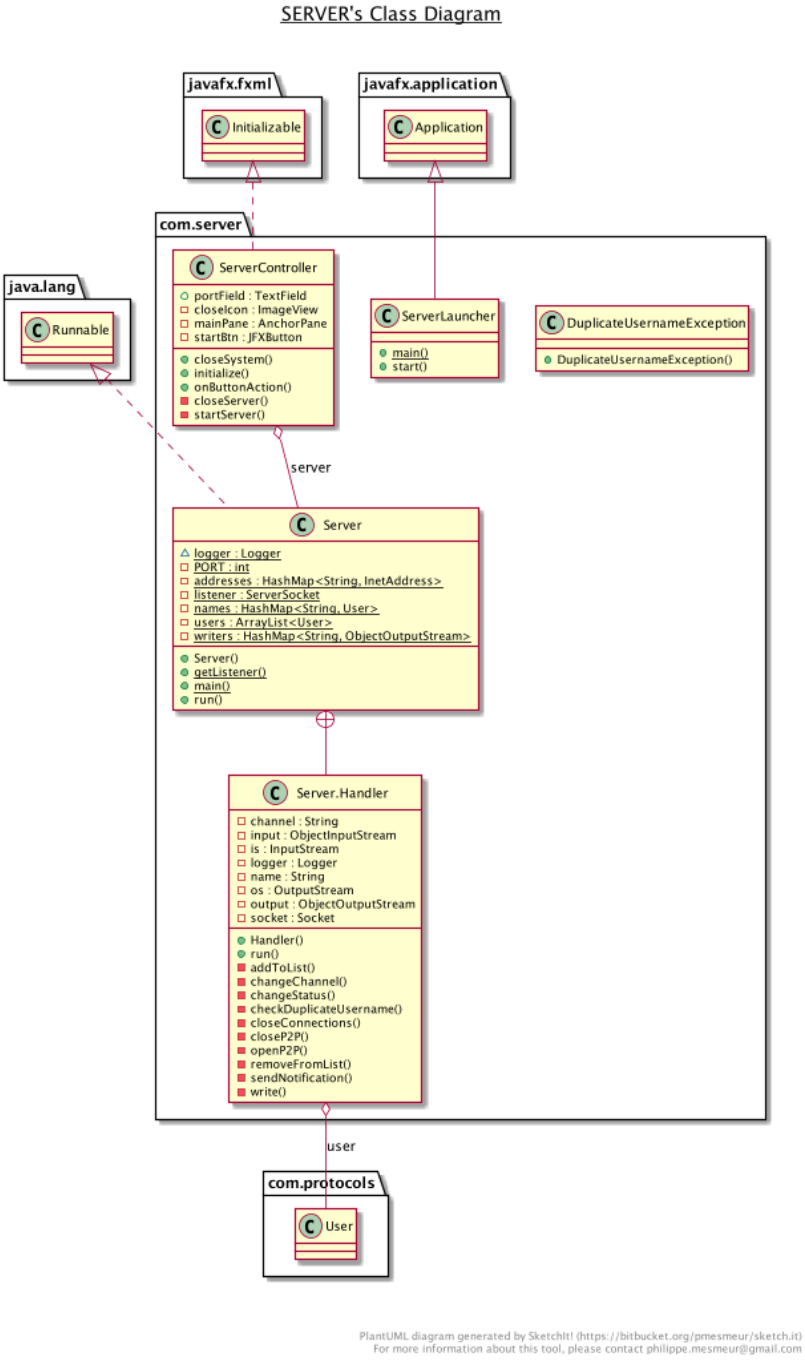
Hình 2: Kiến trúc tổng quát

Như đã trình bày ở phần trên, mỗi client sẽ đăng nhập và chuyển đến phòng chat chung. Nơi hiển thị tin nhắn đến cũng như thực hiện các chức năng cửa ứng dụng.

4.2 Lược đồ lớp

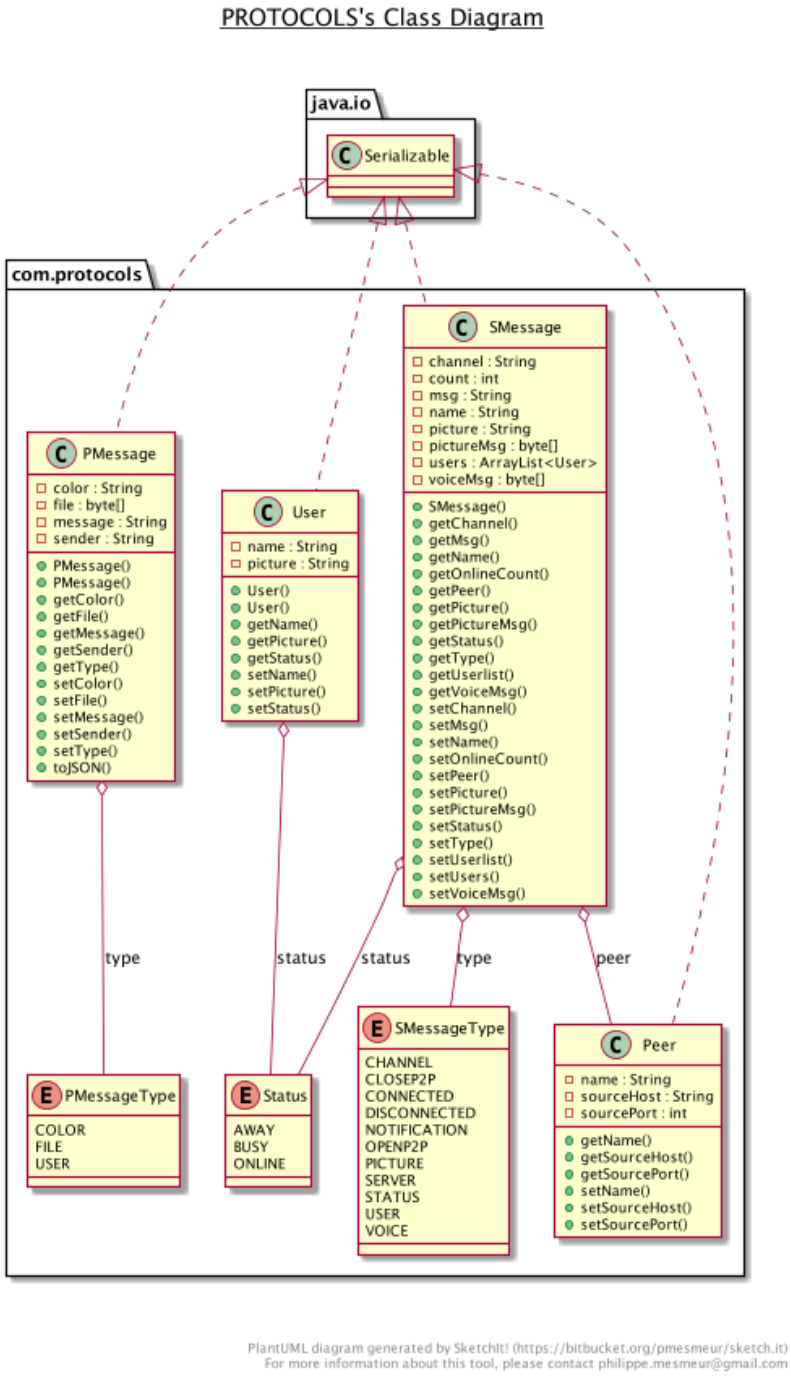
Các package chính trong ứng dụng:

- **com.server** - Gồm các class để hiển thực SocketServer cùng GUI, trong đó class **Server.java** hiện thực để xử lý các yêu cầu chức năng chính.



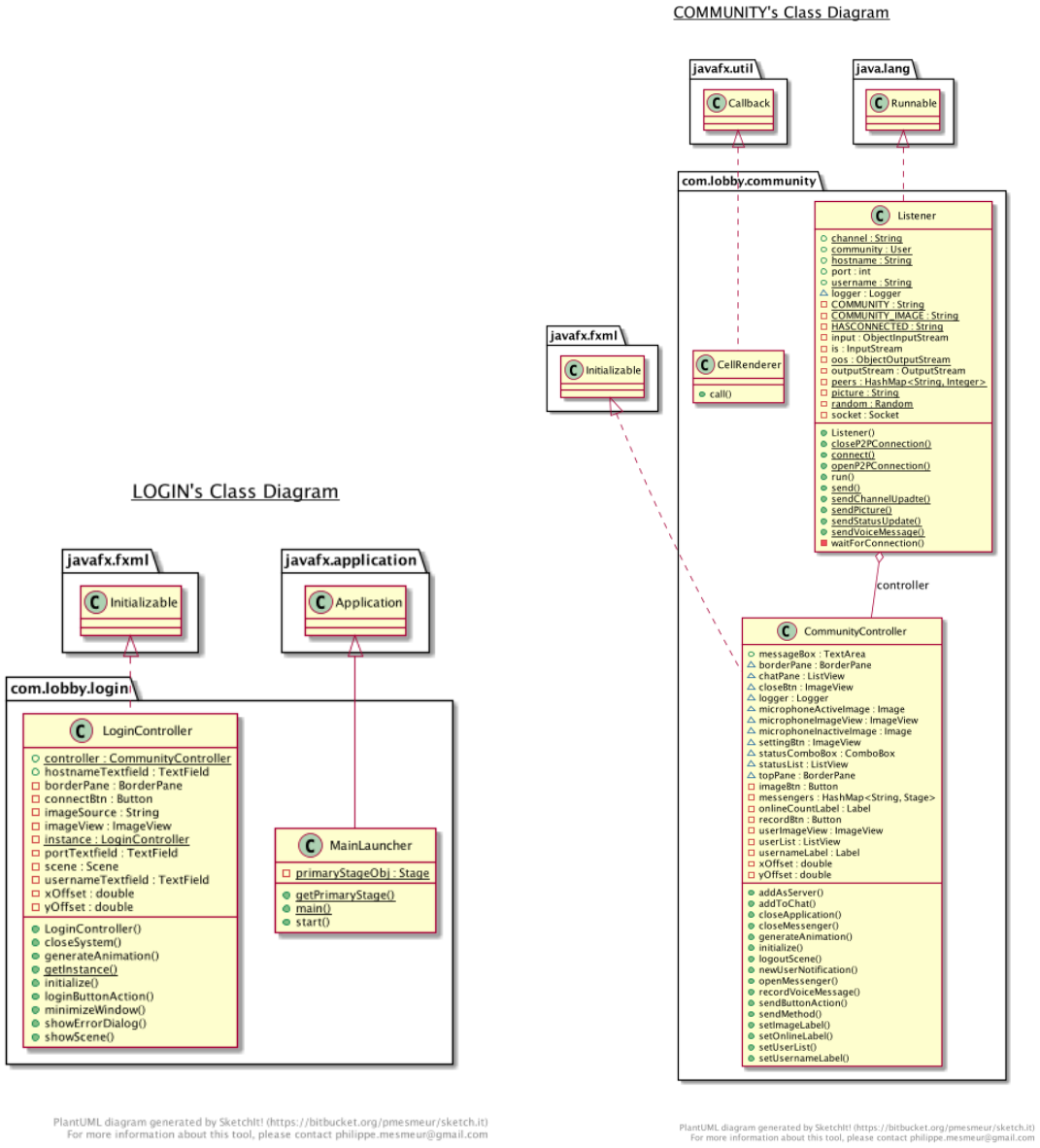
Hình 3: Lược đồ lớp package com.server

- **com.protocols** - Gồm các class định nghĩa các giao thức sử dụng trong ứng dụng, như: **SMessage** - giao thức giao tiếp giữa client-server; **PMessage** - giao thức giao tiếp client-client.



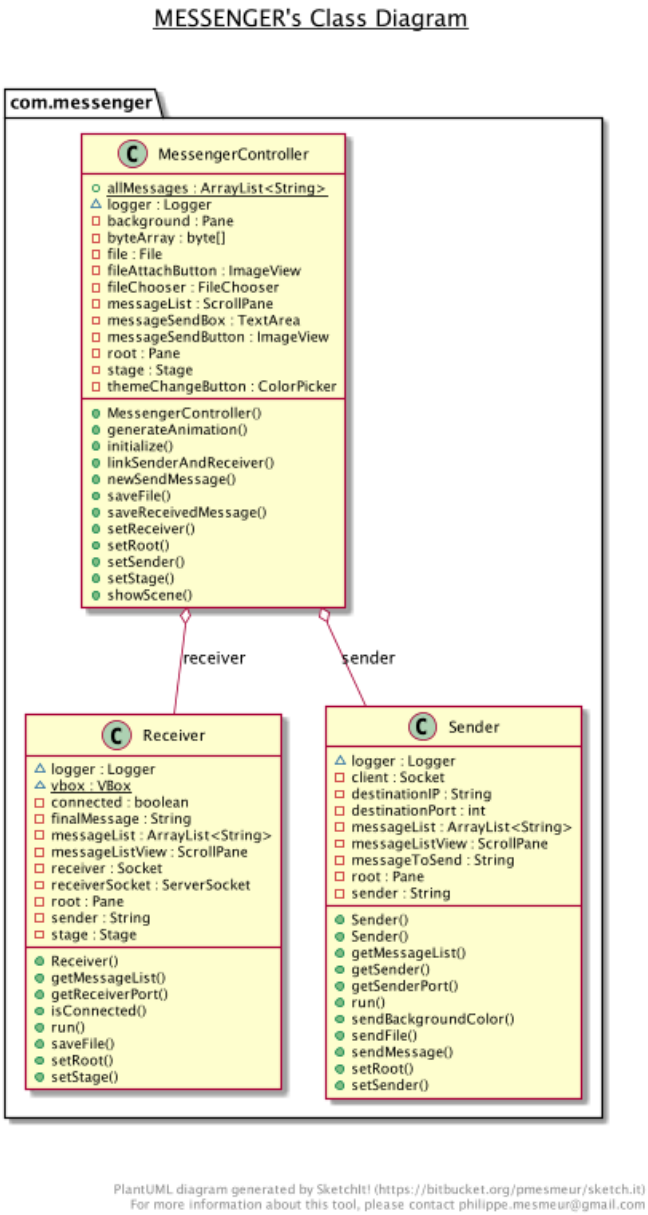
Hình 4: Lược đồ lớp package com.protocols

- **com.lobby** - Gồm các class hiện thực phần front-end, cung cấp cách thức giao tiếp giữa người dùng và server.



Hình 5: Lược đồ các lớp trong package com.lobby

- **com.messenger** - Gồm các class hiện thực giao tiếp trực tiếp giữa client-client.



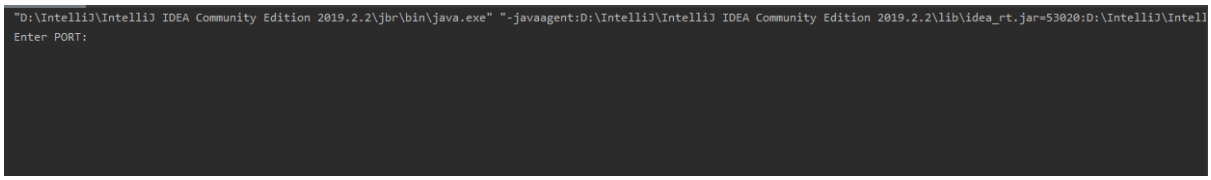
Hình 6: Lược đồ lớp package com.messenger

5 Đánh giá kết quả hiện thực

5.1 Khởi động server

Cách 1: sử dụng server CLI:

- Chạy class com.server.Server.

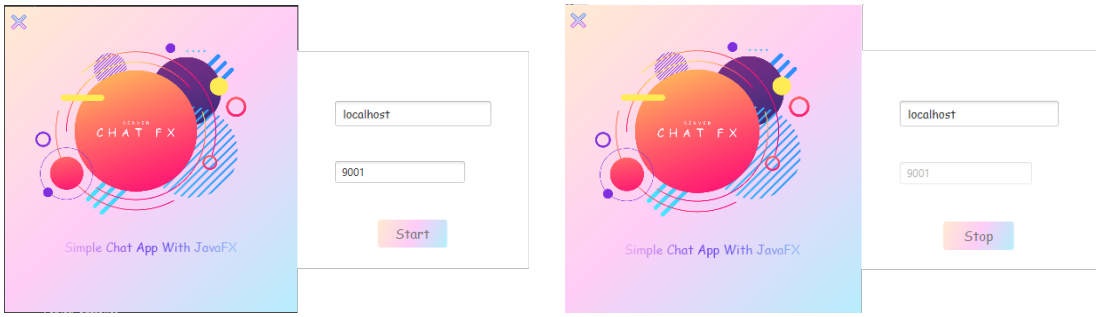


Hình 7: Server CLI

- Nhập tên port vào. VD: 9001.

Cách 2: sử dụng server UI:

- Chạy class com.server.ServerController.



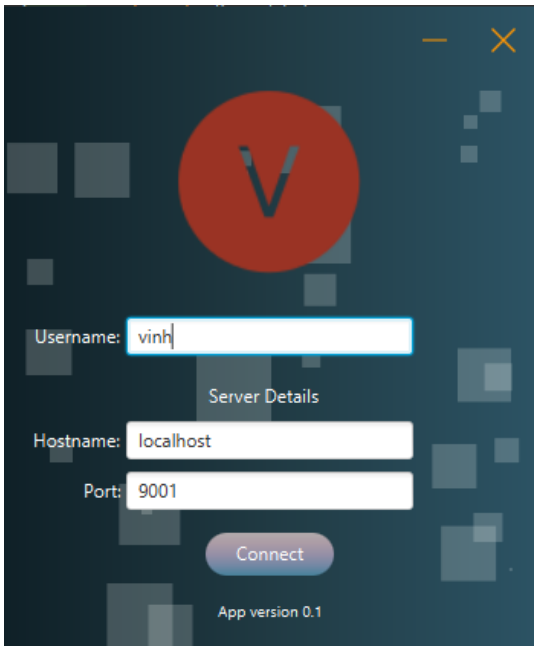
Hình 8: Server UI

- Nhập đường dẫn vào server ở ô localhost. Nhập port vào ô dưới rồi bấm nút Start để khởi động server.
- Để tắt server nhấn nút stop

5.2 Đăng nhập

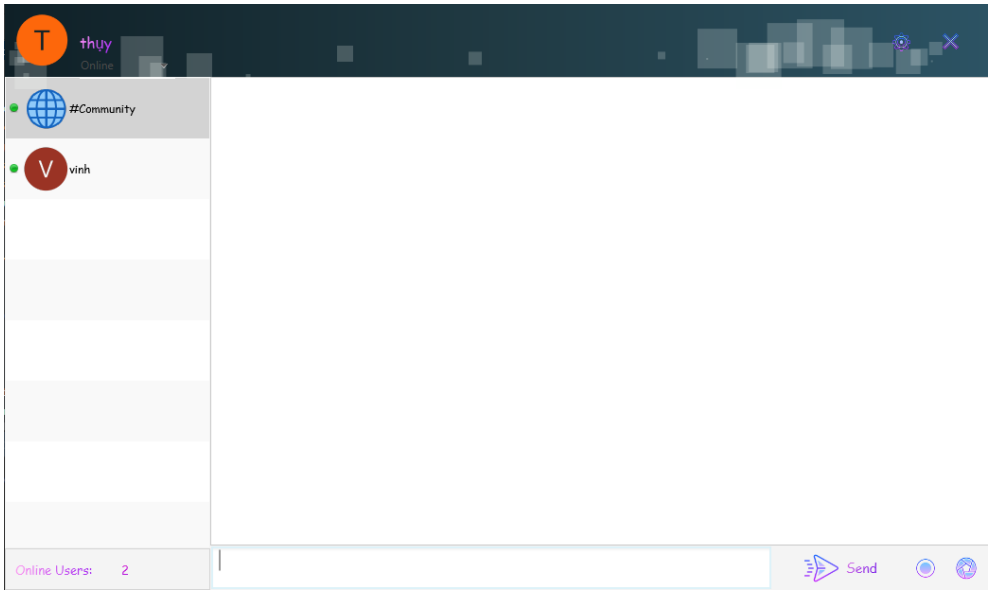
Chạy class com.client.login.Mainlauncher.

- Nhập tên người dùng vào ô username.
- Nhập đường dẫn tới server vào hostname.
- Nhập port kết nối.



Hình 9: Màn hình Login

Giao diện sau khi đăng nhập thành công vào server.



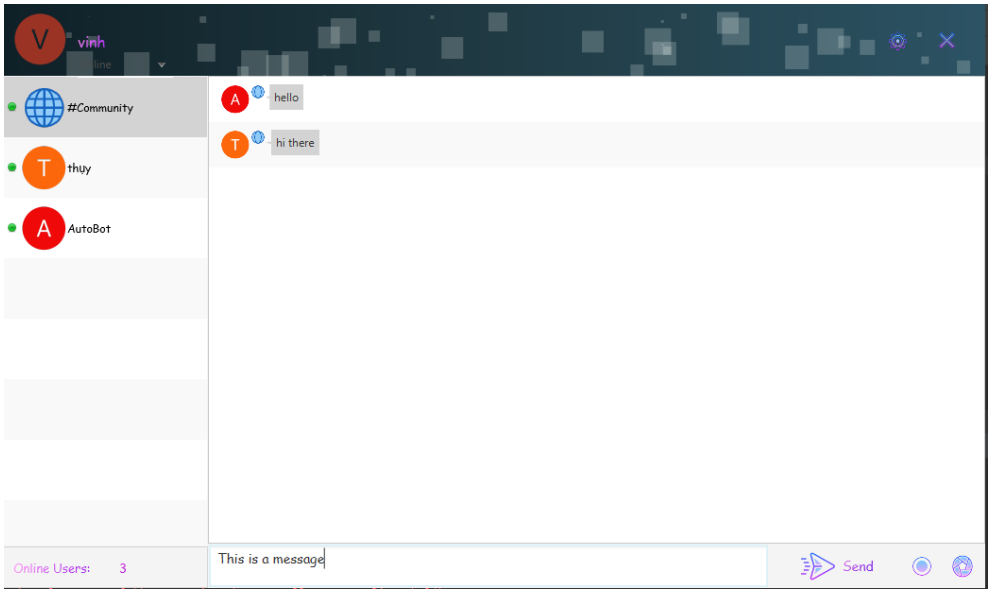
Hình 10: Main Interface

5.3 Gửi tin nhắn

5.3.1 Gửi tin nhắn vào phòng chat chung

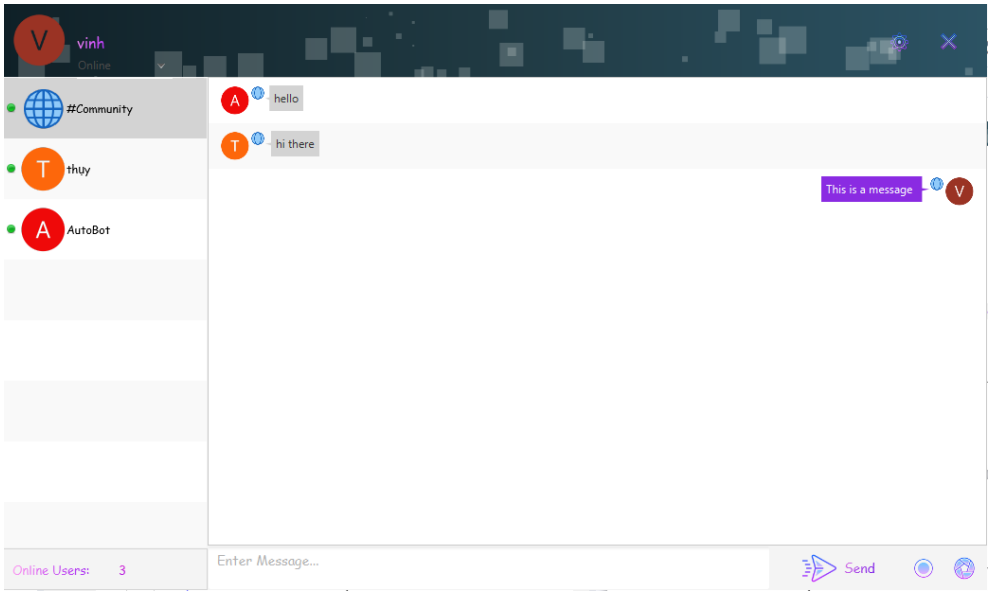
- Nhấn chọn kênh chat community

- Nhập tin nhắn từ bàn phím vào ô chat dưới cùng



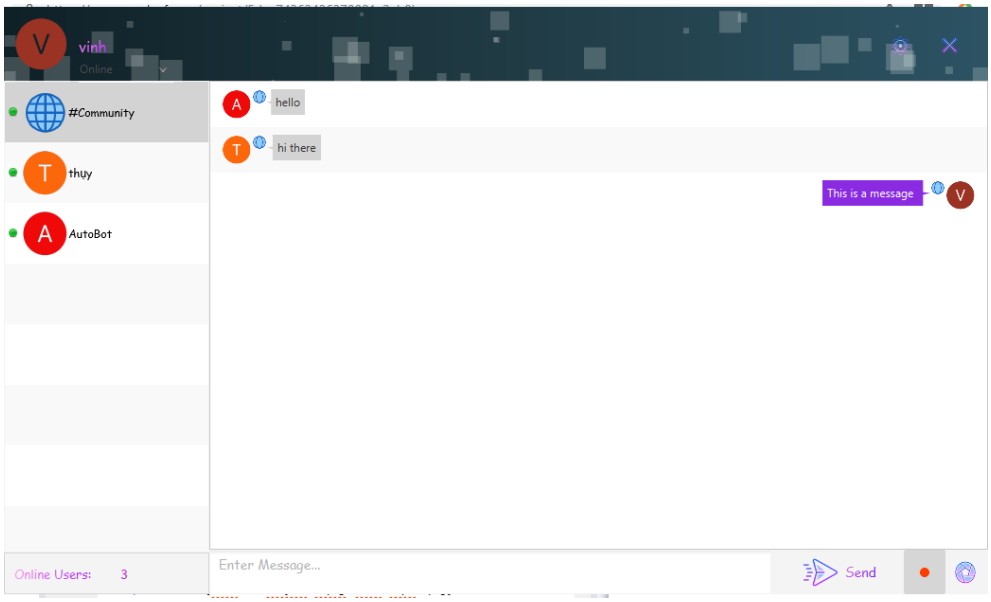
Hình 11: Send public message

- Bấm enter hoặc nhấn nút send cạnh ô chat sẽ được kết quả như hình



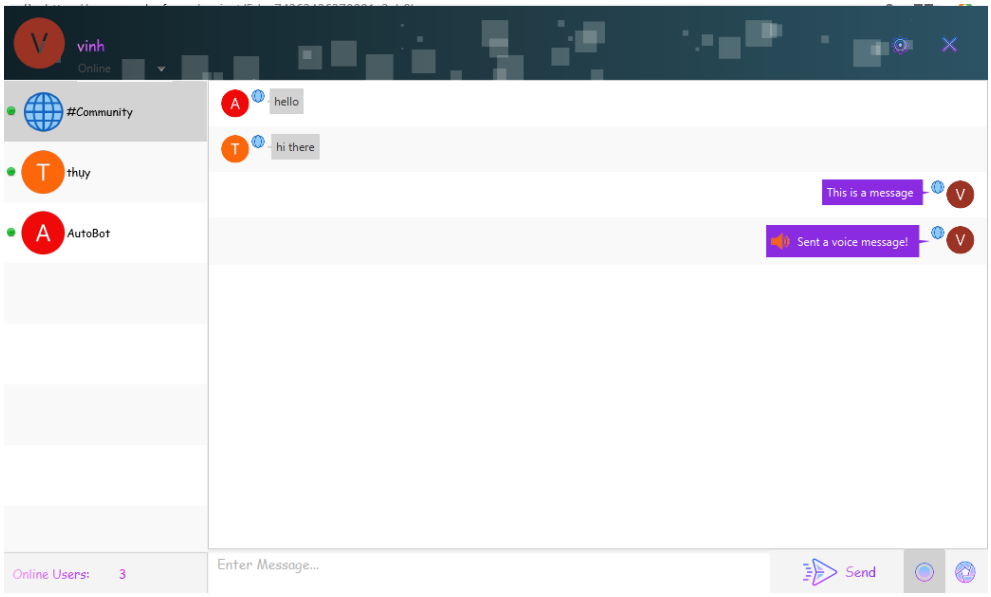
Hình 12: Message sent in community channel

- Để gửi tin nhắn thoại nhấn vào nút thu âm bên phải nút send và bắt đầu nói.



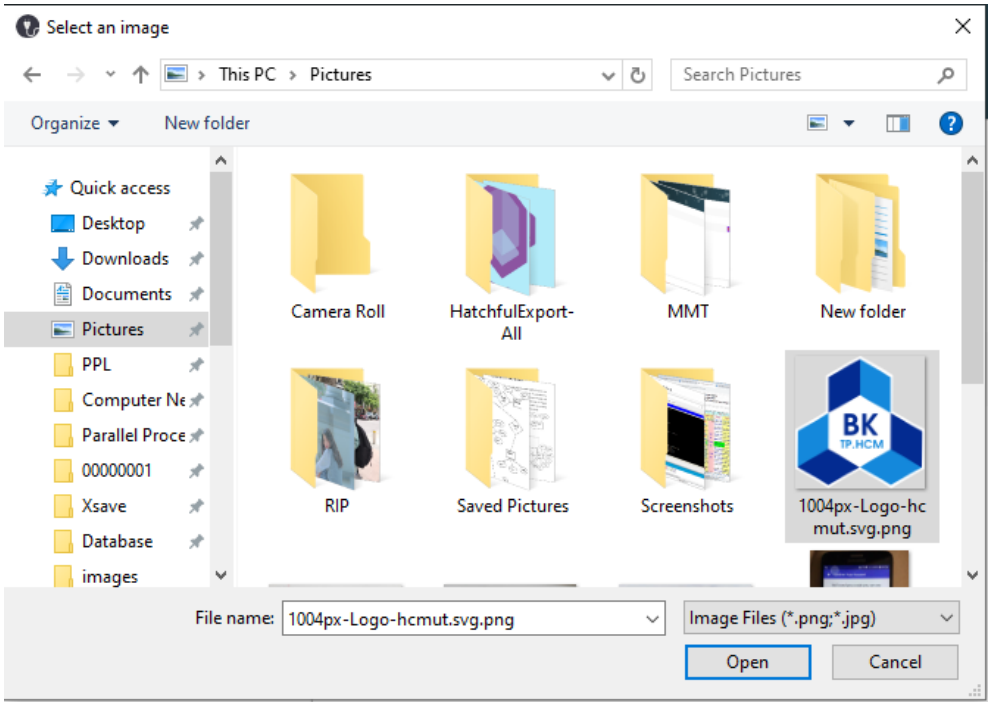
Hình 13: Send voice message

- Nhấn vào nút thu âm 1 lần nữa để gửi đi. Kết quả như hình sau.



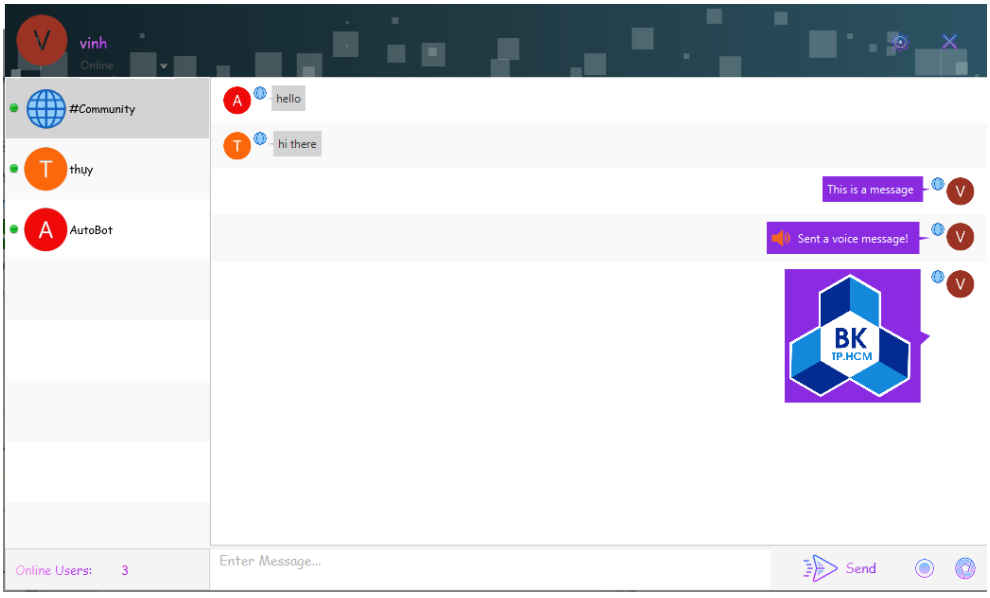
Hình 14: Voice message sent

- Để gửi file nhấn vào nút duyệt file bên phải nút thu âm để mở hộp thoại chọn thư mục.



Hình 15: Browse file

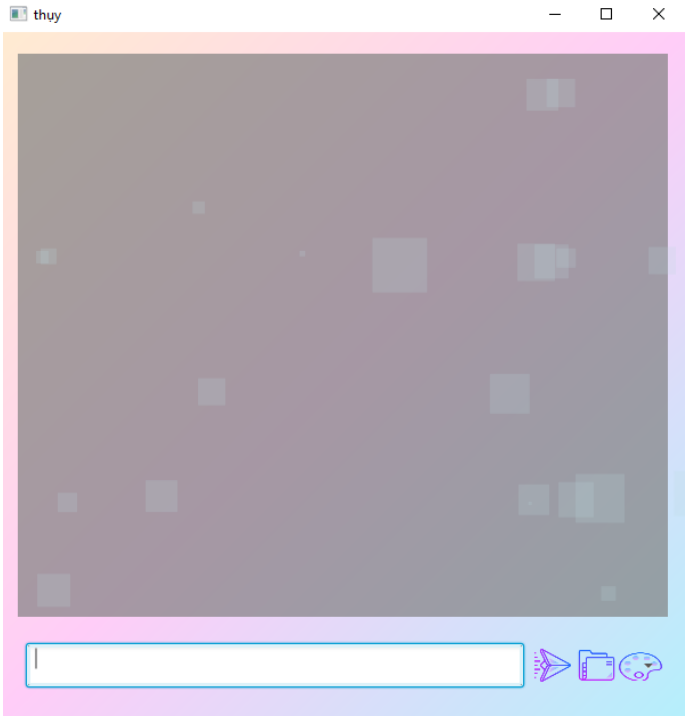
- Nhấn vào file cần chọn. Kết quả như hình sau.



Hình 16: File sent

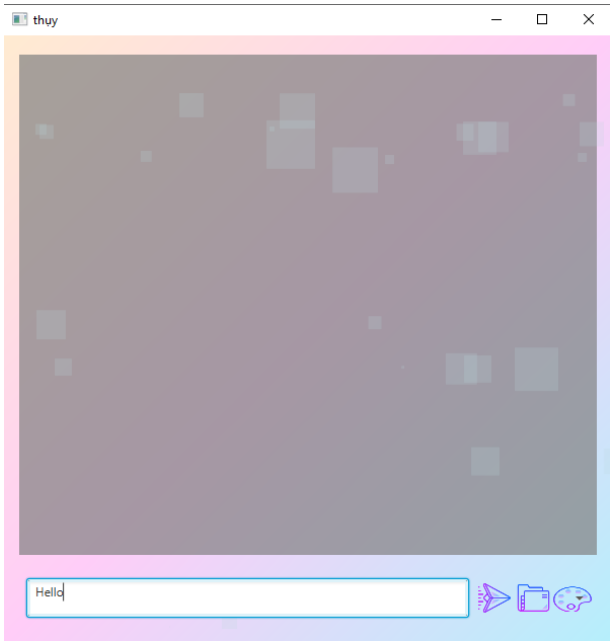
5.3.2 Gửi tin nhắn riêng cho user khác

- Nhấp chọn user để gửi tin nhắn đến.
- Cửa sổ chat mới hiện ra.



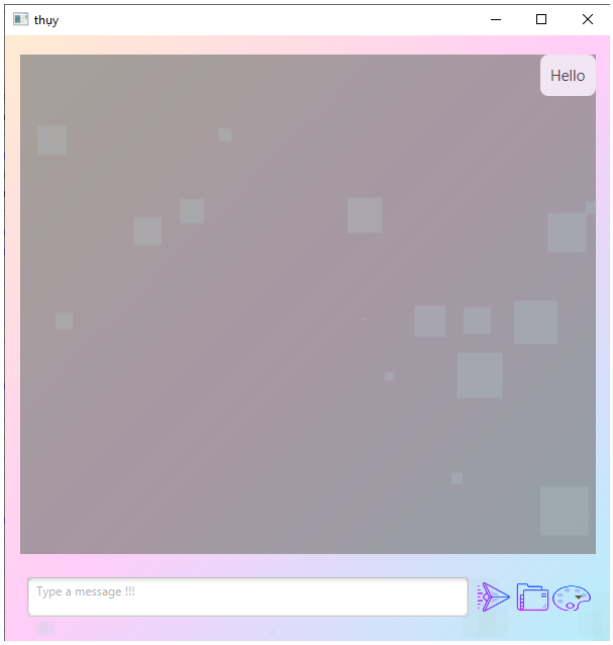
Hình 17: Send private message

- Nhập tin nhắn từ bàn phím vào ô chat dưới cùng.



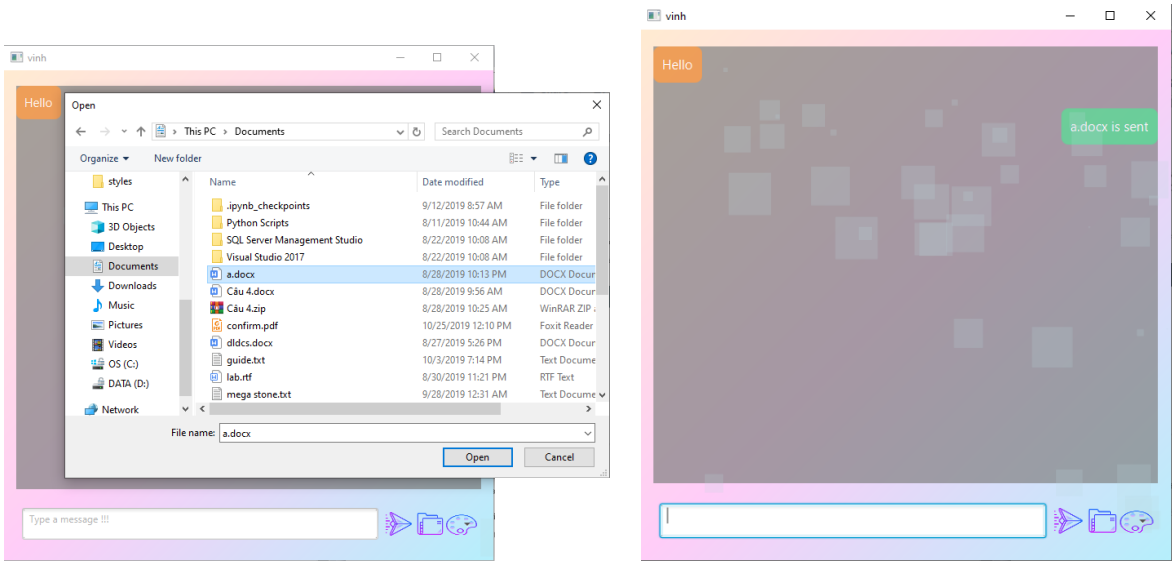
Hình 18: Enter private message

- Nhấn nút send ở bên cạnh ô chat hoặc nhấn phím enter để gửi tin.



Hình 19: Private message sent

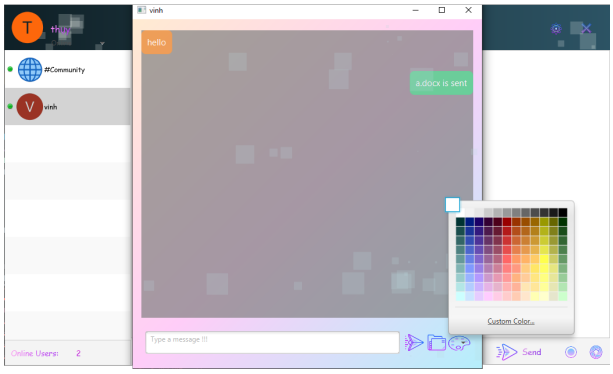
- Để gửi tệp tin nhấn nút hình tệp tin cạnh nút send để mở hộp thoại chọn thư mục.



Hình 20: Browse directory private

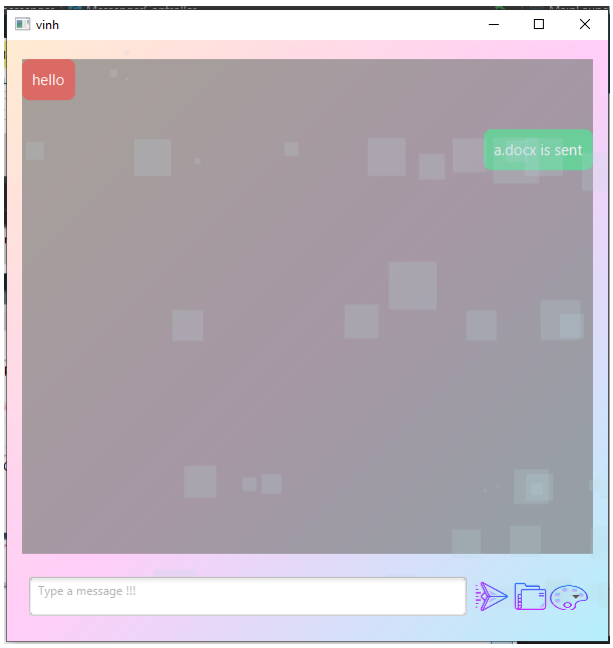
- Nhấn open để gửi

- Để đổi màu bóng chat nhấn chọn nút bảng màu nằm bên phải nút chọn file và chọn màu.



Hình 21: Change chat bubble's color

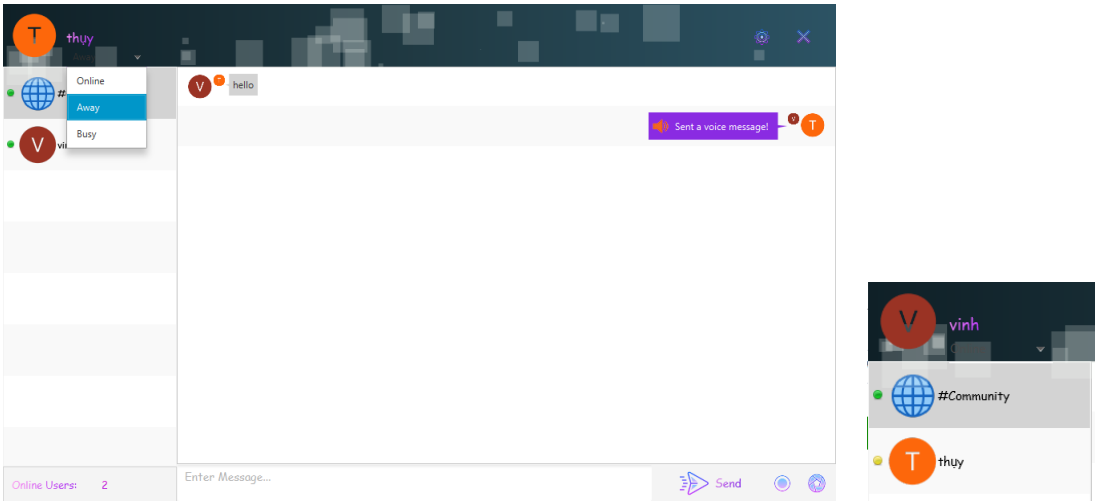
- Kết quả như hình sau.



Hình 22: Chat bubble's color changed

5.4 Chuyển trạng thái hoạt động

Nhấn vào dấu mũi tên ở dưới username góc trên cùng bên trái để mở bảng chuyển trạng thái. Sau



Hình 23: Change state

khi đổi trạng thái, dấu tích ở bên cạnh username sẽ đổi màu đối với các user khác.

Tài liệu

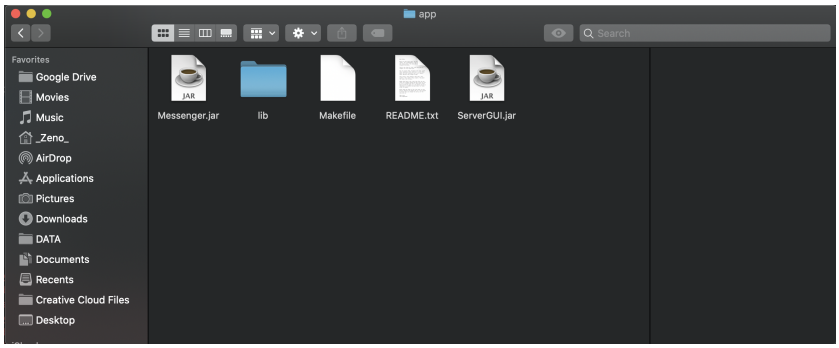
[1] GPCoder *Xây dựng ứng dụng Client-Server với Socket trong Java.*
last visited Oct 28th, 2019.

[2] KeepToo *JavaFX UI design youtube channel* .
Last seen Oct 25th, 2019.

[3] `https://docs.oracle.com/javase/tutorial/`.
Last visited Oct 28th, 2019.

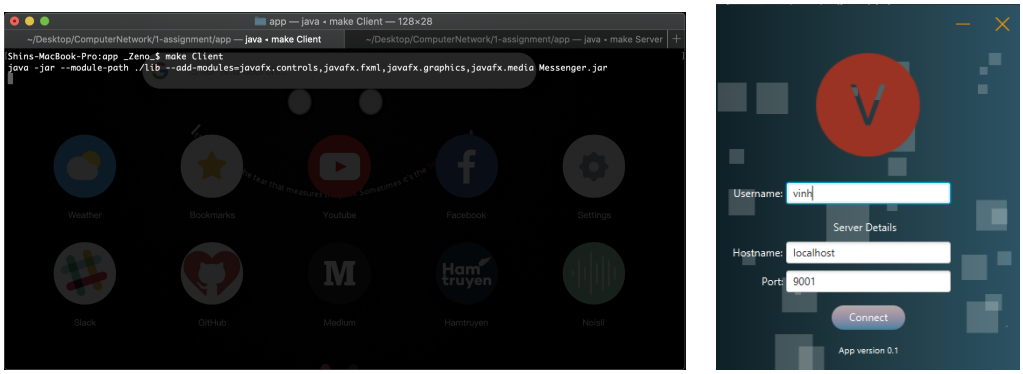
A Hướng dẫn sử dụng

Thư mục *app/* (như hình dưới) chứa các artifacts (file .jar) để mở ứng dụng.



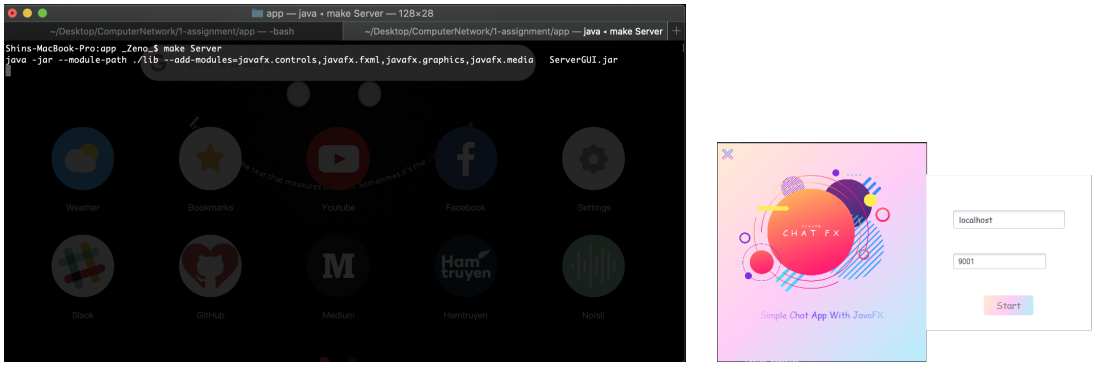
Hình 24: Thư mục app chưa artifacts để mở ứng dụng

Mở terminal trỏ đến thư mục hiện tại và chạy hai dòng lệnh để khởi động ứng dụng `make Client`



Hình 25: Cách mở messenger bằng terminal

Nếu muốn máy tính là một server, chạy dòng lệnh sau: `make Server`



Hình 26: Cách mở server bằng terminal