**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**

**CHUYÊN ĐỀ HỆ THỐNG PHÂN TÁN**

**Thành phố Hồ Chí Minh**

**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**

**| Đề tài |**

**Cài đặt chat GRPC**

**| Giáo viên hướng dẫn |**

**Thầy Dũng Trần Trung**

**Thầy Lê Giang Thanh**

**Thành phố Hồ Chí Minh**

**LỜI CẢM ƠN**

Để có thể hoàn thành đồ án này, chúng em xin gửi lời cảm ơn chân thành tới thầy Dũng Trần Trung, thầy Lê Giang Thanh khoa Công Nghệ Thông Tin, trường Đại học Khoa học Tự nhiên vì đã hỗ trợ tụi em trong suốt quãng thời gian làm đồ án, giúp đồ án này được hoàn thiện nhất có thể.

Do thời gian có hạn cũng như các kiến thức của chúng em vẫn có nhiều hạn chế, rất khó để tụi em tránh khỏi những sai sót khi hoàn thiện đồ án cũng như làm báo cáo. Vì vậy rất mong có thể nhận được những lời góp ý từ các thầy cô để chúng em có thể có thêm kiến thức để hoàn thành những đồ án tiếp theo.

Chúng em xin chân thành cảm ơn.

Mục Lục

[1. Giới thiệu: 7](#_Toc132664937)

[2. Thành viên và bảng phân công: 7](#_Toc132664938)

[3. Cách chạy chương trình: 7](#_Toc132664939)

[4. Cách dùng: 8](#_Toc132664940)

[5. Mô tả chi tiết hệ thống: 10](#_Toc132664941)

[6. Chức năng chi tiết của từng Class: 10](#_Toc132664942)

[a) class ChatServiceGrpc 10](#_Toc132664943)

[b) class ChatServer 11](#_Toc132664944)

[c) class ChatClient 11](#_Toc132664945)

[d) class ChatServiceImpl 12](#_Toc132664946)

**BÁO CÁO**

# Giới thiệu:

* Chương trình chạy trên ngôn ngữ Java và dùng GRPC.
* Gồm có Server và Client. Khởi chạy Sever trước sau đó chạy những Client thì các Client sẽ kết nổi được với nhau.
* Mỗi Client sẽ gồm có 3 chức năng (Gửi tin nhắn, like, disconnect).
* Khi 1 Client gửi tin nhắn thì sẽ được broadcast cho những Client còn lại thấy tin nhắn.
* Các Client khác có thể like tin nhắn đó. Và Client sau khi gửi tin nhắn thì tin nhắn đó cần có ít nhất 2 like trở lên thì mới có thể nhắn tin tiếp theo.l
* Chương trình có thể chạy với số Client không giới hạn, phụ thuộc vào phần cứng.

# Thành viên và bảng phân công:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Nhiệm vụ** |
| 1 | Nguyễn Gia Minh | 19126054 | * Làm báo cáo. * Xử lý thuật toán. * Gửi nhận buffer và in ra tin nhắn. * Xử lý Vector Clock và gửi nhận thông tin đồng bộ hóa. * Clean code. |
| 2 | Lê Trung Hiếu | 19126044 | * Xử lý thuật toán. * Xử lý đầu ra đầu vào. * Xử lý gửi nhận thông tin qua process và thread . * Lọc data. * Validate code. |

# Cách chạy chương trình:

* Đầu tiên chạy file ChatServer.java
* Sau đó chạy file ChatClient.java
* Ở cmd ChatClient.java thì điền thông tin như sau:
* Name: tùy chọn
* Server address: localhost
* Server port: 50051
* Nếu như thấy thông báo “Connected to ChatServer” thì có nghĩa là Client đã kết nối được với Server.

Graphical user interface, application

Description automatically generated

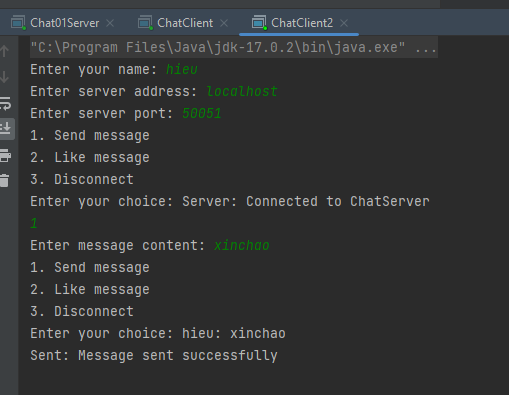
Text

Description automatically generated

# Cách dùng:

Chức năng 1 (Send message):

* Khi nhấn 1 và enter thì ta sẽ gõ tin nhắn
* Nếu tin nhắn được gửi broadcast cho tất cả Client thành công thì sẽ có thông báo “Message sent successfully”
* Và bên phía Client còn lại sẽ thấy được tin nhắn như hình dưới đây.



Text

Description automatically generated

Chức năng 2 (Like message):

* Khi bấm 2 và enter, ta sẽ nhập theo cú pháp sender + content.
* Ví dụ dưới đây là like message của “hieu” và nội dung tin nhắn là “xinchao”
* Nếu nhận được thông báo là “Message liked successfully” có nghĩa là đã like thành công.

Text

Description automatically generated

Chức năng 3 (Disconnect):

* Khi chọn chức năng này Client đó sẽ thoát khỏi cuộc trò chuyện và tất cả Client khác sẽ nhận được thông báo như sau.

Graphical user interface, text, application

Description automatically generated

# Mô tả chi tiết hệ thống:

Chương trình này dùng gRPC framework để tạo các giao tiếp giữa client và server.

User.proto: Đây là file định nghĩa giao diện và dữ liệu cho hệ thống chat gRPC. File này được sử dụng để build mã nguồn Java sau khi được biên dịch bằng Protocol Buffers (protobuf) compiler từ các Dependencies được import trong pom.xml.

Nội dung của file đó được tổ chức gồm các class chính dưới đây:

* **ChatServiceGrpc**: là giao thức định nghĩa các phương thức mà hệ thống chat hỗ trợ, bao gồm sendMessage, likeMessage, getMessages và registerUser. Mỗi phương thức có các tham số đầu vào và giá trị trả về tương ứng.
* **ChatServer** là server, được triển khai dựa trên giao diện ChatService được định nghĩa trong file User.proto.
* **ChatClient:** Class này được sử dụng trong việc để gửi tin nhắn, like tin nhắn, và đăng ký người dùng với ChatServer.
* **ChatServiceImpl:** dùng để implement các phương thức quan trọng như sendMessage(), likeMessage(), getMessages().

# Chức năng chi tiết của từng Class:

## class ChatServiceGrpc

Đây là các định nghĩa cho các kiểu dữ liệu được sử dụng trong các phương thức của giao diện ChatService, bao gồm Message (đại diện cho một tin nhắn), User (đại diện cho một người dùng), và Like (đại diện cho một lượt like).

## class ChatServer

ChatServer có 1 vài thuộc tính như:

* **port:** là số cổng được sử dụng bởi server.
* **server:** là một đối tượng Server của gRPC được tạo bằng ServerBuilder và được sử dụng để khởi động server.
* **clients:** là một ConcurrentHashMap để lưu trữ các client đang kết nối với server, với key là tên của client và value là đối tượng StreamObserver<User.ChatMessage> được sử dụng để gửi các tin nhắn tới client đó.
* **messageLikes:** là một ConcurrentHashMap để lưu trữ số lượng lượt thích của mỗi tin nhắn được gửi đến server.
* **main():** Đây là phương thức chạy chính của dịch vụ chat server, nơi mà server được khởi tạo và lắng nghe các kết nối từ các client.
* **start():** Phương thức này được gọi trong phương thức main() để bắt đầu hoạt động của server. Nó khởi tạo một server gRPC, đăng ký dịch vụ ChatServiceImpl (được triển khai từ giao diện ChatService), và bắt đầu lắng nghe các kết nối từ các client.
* **shutdown():** Phương thức này được gọi để đóng server gRPC khi không còn cần thiết.
* **broadcastMessage():** Phương thức này được gọi để phát tin nhắn từ một người dùng đến tất cả các người dùng khác trong nhóm chat.

## class ChatClient

* **main():** Đây là phương thức chạy chính của dịch vụ chat client, nơi mà client được khởi tạo và kết nối với dịch vụ chat server.
* **start():** Phương thức này được gọi trong phương thức main() để bắt đầu hoạt động của client. Nó khởi tạo một kết nối gRPC đến dịch vụ chat server, và cho phép người dùng nhập các lệnh để gửi tin nhắn, like tin nhắn, và đăng ký người dùng.

* **sendMessage():** Phương thức này được gọi để gửi một tin nhắn từ client đến dịch vụ chat server.
* **likeMessage():** Phương thức này được gọi để gửi một yêu thích (like) tin nhắn từ client đến dịch vụ chat server.
* **registerUser():** Phương thức này được gọi để đăng ký một người dùng mới với dịch vụ chat server.
* **shutdown():** Phương thức này được gọi để đóng kết nối gRPC và kết thúc hoạt động của client. class BufferModel

## class ChatServiceImpl

Trong class ChatServiceImpl này:

* **sendMessage():** sẽ được gọi khi một client gửi tin nhắn đến server. Trước khi phát sóng tin nhắn này tới tất cả các client kết nối, phương thức này sẽ kiểm tra xem người gửi đã gửi tin nhắn được thích hợp hay chưa. Nếu tin nhắn đã được gửi đến ít nhất hai lần, nó sẽ được phát sóng đến tất cả các client kết nối. Nếu không, server sẽ trả về một tin nhắn không thành công cho người gửi.
* **likeMessage():** khi một client yêu cầu tăng một like cho một tin nhắn, server sẽ kiểm tra xem tin nhắn đã được đăng ký với server hay chưa. Nếu có, nó sẽ tăng giá trị của like hiện tại cho tin nhắn và kiểm tra nếu giá trị này đạt đến hoặc vượt qua giới hạn, nó sẽ xóa tin nhắn khỏi bộ đếm.
* **receiveMessage():** sẽ được gọi khi một client kết nối với server. Server sẽ lưu trữ một danh sách các client kết nối và sẽ thông báo cho client rằng kết nối đã được thiết lập thành công. Sau đó, nó sẽ tiếp tục lắng nghe tin nhắn từ client và không ngừng lặp lại quá trình này.