# **MỤC LỤC**

[**MỤC LỤC** a](#_Toc505355881)

[**TÓM TẮT** d](#_Toc505355882)

[**DANH MỤC HÌNH ẢNH** e](#_Toc505355883)

[**DANH MỤC BẢNG BIỂU** e](#_Toc505355884)

[**I-** **Giới thiệu đề tài.** 1](#_Toc505355885)

[**1.** **Đặt vấn đề.** 1](#_Toc505355886)

[**2.** **Mục tiêu đề tài.** 1](#_Toc505355887)

[**3.** **Phạm vi đề tài.** 2](#_Toc505355888)

[**4.** **Cấu trúc báo cáo.** 2](#_Toc505355889)

[**II-** **Công nghệ ứng dụng, giải pháp, thuật toán, hiện thực hệ thống.** 3](#_Toc505355890)

[**1.** **Công nghệ ứng dụng** 3](#_Toc505355891)

[***1.1*** ***Sơ lược về ứng dụng client – server*** 3](#_Toc505355892)

[***1.2*** ***Các khái niệm mạng máy tính cơ bản*** 3](#_Toc505355893)

[**1.2.1** **Các loại giao thức mạng phổ biến** 3](#_Toc505355894)

[***1.3*** ***Socket, port và Java socket*** 5](#_Toc505355895)

[**1.3.1** **Socket** 5](#_Toc505355896)

[**1.3.2** **Port** 6](#_Toc505355897)

[**1.3.3** **Java socket** 6](#_Toc505355898)

[***1.4*** ***Đa luồng (multi-threading)*** 7](#_Toc505355899)

[**1.4.1** **Đa luồng là gì** 7](#_Toc505355900)

[**1.4.2** **Java Thread** 8](#_Toc505355901)

[***1.5*** ***JSON*** 8](#_Toc505355902)

[**1.5.1** **Giới thiệu về JSON.** 8](#_Toc505355903)

[**1.5.2** **Kiểu dữ liệu của JSON** 9](#_Toc505355904)

[**1.5.3** **Ứng dụng của chuỗi JSON** 9](#_Toc505355905)

[**2.** **Phân tích chức năng hệ thống** 10](#_Toc505355906)

[***2.1*** ***Mô tả kiến trúc phần mềm.*** 10](#_Toc505355907)

[***2.2*** ***Chức năng chi tiết*** 10](#_Toc505355908)

[***2.2.1*** ***Chức năng của server*** 10](#_Toc505355909)

[***2.2.2*** ***Chức năng của client*** 11](#_Toc505355910)

[***2.2.3*** ***Use Case Diagram*** 12](#_Toc505355911)

[***2.2.4*** ***Use Case Scenario*** 12](#_Toc505355912)

[***2.2.5*** ***Tổng quan về cấu trúc cơ sở dữ liệu của hệ thống.*** 14](#_Toc505355913)

[***2.2.6*** ***Mô hình cấu trúc cơ sở dữ liệu.*** 14](#_Toc505355914)

[**3.** **Hiện thực hệ thống.** 16](#_Toc505355915)

[***3.1*** ***Giao thức của ứng dụng*** 16](#_Toc505355916)

[***3.1.1*** ***Cấu trúc cơ bản của một thông điệp*** 16](#_Toc505355917)

[***3.1.2*** ***Giao thức các thông điệp gửi từ phía client*** 17](#_Toc505355918)

[***3.1.3*** ***Giao thức các thông điệp gửi từ phía server*** 19](#_Toc505355919)

[***3.2*** ***Sơ đồ class của ứng dụng*** 25](#_Toc505355920)

[***3.2.1*** ***Sơ đồ class tổng quát*** 25](#_Toc505355921)

[***3.2.2*** ***Sơ đồ class cho phần dùng chung mics*** 26](#_Toc505355922)

[***3.2.3*** ***Sơ đồ class cho phía server*** 28](#_Toc505355923)

[***3.2.4*** ***Sơ đồ class phía Client*** 32](#_Toc505355924)

[**III-** **Kết quả.** 36](#_Toc505355925)

[**1.** **Giao diện ứng dụng.** 36](#_Toc505355926)

[***1.1*** ***Server.*** 36](#_Toc505355927)

[***1.2*** ***Client.*** 38](#_Toc505355928)

[***1.2.1*** ***Màn hình chính*** 38](#_Toc505355929)

[***1.2.2*** ***Màn hình trò chơi*** 42](#_Toc505355930)

[**2.** **Đánh giá hệ thống.** 46](#_Toc505355931)

[***2.1*** ***Ưu điểm.*** 46](#_Toc505355932)

[***2.2*** ***Nhược điểm.*** 46](#_Toc505355933)

[**IV-** **Kết luận.** 46](#_Toc505355934)

[**1.** **Kết luận.** 46](#_Toc505355935)

[**2.** **Hướng phát triển.** 46](#_Toc505355936)

# **TÓM TẮT**

Đề tài thực tập của nhóm chúng tôi là xây dựng ứng dụng mạng là trò chơi “Kéo búa bao”. Ứng dụng này dựa trên mô hình “client – server” truyền tải thông tin dựa trên giao thức TPC/IP cùng với các công nghệ Java Socket, Thread, JSON và một protocol được tạo riêng để quản lý các thông điệp qua lại giữa hai phía. Trong quá trình thực hiện đề tài thực tập này, nhóm chúng tôi đã thực hiện thông qua hai giai đoạn cụ thể sau:

Giai đoạn đầu: Tiến hành nghiên cứu cơ sở lý thuyết và các công nghệ để phân tích, đánh giá và đề ra những chức năng cần được phát triển cũng như phương pháp thực hiện.

Giai đoạn sau: Tiến hành viết ứng dụng hoàn chỉnh dựa trên những chức năng đã đề ra. Theo đó, Những chức năng được phát triển bao gồm:

+ Chức năng đăng ký, đăng nhập, đăng xuất.

+ Chức năng thách đấu và tham gia trận đấu.

Nội dung bài báo cáo này gồm 5 phần:

* 1. Giới thiệu đề tài: Nội dung phần này sẽ tập trung giới thiệu tổng quan về đề tài, làm rõ mục tiêu, phạm vi đề tài và giới thiệu cấu trúc báo cáo.
* 2. Công nghệ ứng dụng, giải pháp, thuật toán, hiện thực hệ thống: Phần này là thân bài chính của bài báo cáo. Nội dung sẽ tập trung trình bày những kiến thức cần thiết cho quá trình thực hiện đề tài gồm những kiến thức tổng quan về Socket, port, Java Socket library, Thread, JSON. Bên cạnh đó đưa ra những phân tích cũng như cách thức hiện thực ứng dụng.
* 3 Kết quả: Nội dung của phần này là đưa ra kết quả cuối cùng của quá trình hiện thực. Hiển thị cách thức hoạt động, luồng chạy cơ bản của ứng dụng.
* 4. Kết luận: Nội dung của phần này là đánh giá quá trình hiện thực đề tài, nhìn nhận những mặt hạn chế và đưa ra những phương hướng phát triển, mở rộng đề tài và đưa vào thực tế.

# **DANH MỤC HÌNH ẢNH**

[Hình 1 Cấu trúc cơ bản của một chuỗi JSON 8](#_Toc505254216)

[Hình 2 Sơ đồ Use Case của ứng dụng 12](#_Toc505254217)

[Hình 3 Mô hình cấu trúc cơ sở dữ liệu 15](#_Toc505254218)

[Hình 4 Cấu trúc class tổng quan của ứng dụng 25](#_Toc505254219)

[Hình 5 Sơ đồ class của package onetwothree.misc 26](#_Toc505254220)

[Hình 6 Sơ đồ class của package onetwothree.server 28](#_Toc505254221)

[Hình 7 Sơ đồ class của package onetwothree.client 32](#_Toc505254222)

[Hình 8 Giao diện bắt đầu của server 36](#_Toc505254223)

[Hình 9 Giao diện nhật ký của server khi có người dùng đăng nhập / đăng xuất 37](#_Toc505254224)

[Hình 10 Giao diện đăng nhập thành công của client 38](file:///D:\www\OTT\Đồ%20án%2020180125.docx#_Toc505254225)

[Hình 11 Giao diện đăng nhập của client 38](file:///D:\www\OTT\Đồ%20án%2020180125.docx#_Toc505254226)

[Hình 12 Giao diện client sau khi có người dùng mới đăng nhập 39](#_Toc505254227)

[Hình 13 Bảng hỏi địa chỉ IP của server 39](file:///D:\www\OTT\Đồ%20án%2020180125.docx#_Toc505254228)

[Hình 14 Giao diện đăng thành công của client 40](file:///D:\www\OTT\Đồ%20án%2020180125.docx#_Toc505254229)

[Hình 15 Giao diện đăng ký của client 40](file:///D:\www\OTT\Đồ%20án%2020180125.docx#_Toc505254230)

[Hình 16 Giao diện đăng xuất của client 41](#_Toc505254231)

[Hình 17 Giao diện thông báo có người thách đấu 42](#_Toc505254232)

[Hình 18 Giao diện thông báo từ chối thách đấu 43](#_Toc505254233)

[Hình 19 Giao diện trò chơi 44](#_Toc505254234)

[Hình 20 Giao diện khi trò chơi kết thúc 45](#_Toc505254235)

# **DANH MỤC BẢNG BIỂU**

[Bảng 1 - Mô tả thuộc tính class MessageHandler 26](#_Toc505254648)

[Bảng 2 - Mô tả các phương thức class MessageHandler 27](#_Toc505254649)

[Bảng 3 - Mô tả thuộc tính class OneTwoThreeServer 29](#_Toc505254650)

[Bảng 4 - Mô tả phương thức class OneTwoThreeServer 29](#_Toc505254651)

[Bảng 5 - Mô tả thuộc tính class Authentication 30](#_Toc505254652)

[Bảng 6 - Mô tả phương thức class Authentication 30](#_Toc505254653)

[Bảng 7 - Mô tả các thuộc tính class Game 31](#_Toc505254654)

[Bảng 8 - Mô tả phương thức class Game 32](#_Toc505254655)

[Bảng 9 - Mô tả thuộc tính class OneTwoThreeClient 33](#_Toc505254656)

[Bảng 10 - Mô tả phương thức class OneTwoThreeClient 34](#_Toc505254657)

[Bảng 11 - Mộ tả thuôc tính class ClientGameBoard 34](#_Toc505254658)

[Bảng 12 - Mô tả phương thức class ClientGameBoard 35](#_Toc505254659)

1. **Giới thiệu đề tài**
2. **Đặt vấn đề.**

Từ khi ra đời cho đến nay, mạng máy tính đã và đang là một phần không thể thiếu cho xã hội hiện đại. Những ứng dụng của mạng máy tính vô cùng đa dạng và được áp dụng trên rất nhiều lĩnh vực của đời sống, là một trong những nhân tố bản lề, làm bàn đạp cho sự phát triển của xã hội.

Điểm đặt biệt của ứng dụng mạng so với các ứng dụng khác là nó cho phép các máy tính khác nhau trong cùng một mạng có thể cài đặt cùng một ứng dụng, và thông qua ứng dụng đó, những máy tính này có thể giao tiếp với nhau để cùng thực hiện một công việc chung, một mục đích chung nào đó.

Một ứng dụng mạng điển hình có thể kể đến là ứng dụng chat qua mạng Skype. Người dùng cài đặt ứng dụng này, và đăng nhập vào hệ thống với tài khoản được đăng ký trước, sau đó có thể tự do trò chuyện, trao đổi thông tin qua mạng với các người dùng khác.

Câu hỏi được đặt ra là bằng cách nào các ứng dụng mạng có thể trao đổi được thông tin? Người dùng trong các ứng dụng này được quản lý như thế nào? Các công nghệ gì đã được sử dụng?

Với mục đích có được hiểu biết nền tảng về mạng máy tính, trong bài thực tập tốt nghiệp này, chúng tôi sẽ tiến hành xây dựng một ứng dụng mạng đơn giản là trò chơi “Kéo Búa Bao dựa trên mô hình client - server”, thông qua đó có được một cái nhìn rõ hơn về mạng máy tính và cách một ứng dụng mạng hoạt động.

1. **Mục tiêu đề tài.**

Các mục tiêu của đề tài:

* Hiểu một cách tổng quát các lý thuyết về mạng
* Cách thức để các ứng dụng mạng giao tiếp với nhau.
* Các công nghệ nào được sử dụng để giúp cho việc giao tiếp này diễn ra dễ dàng và hợp lý.
* Xây dựng một ứng dụng trò chơi “Kéo búa bao”
* Cho phép người dùng có thể đăng ký tài khoản, đăng nhập vào hệ thống
* Các người dùng đăng nhập vào hệ thống có thể sử dụng ứng dụng trò chơi để tham gia trận đấu với nhau

Kiến trúc hệ thống gồm:

* Một ứng dụng server để nhận các yêu cầu của người chơi và đáp ứng các yêu cầu đó như quản lý người dùng, quản lý các thao tác đăng nhập, đăng xuất, đăng ký tài khoản, thông báo các sự kiện tới người chơi. Ứng dụng server phải kết nối với database để có thể quản lý được người dùng một cách dễ dàng.
* Một ứng dụng client cho phép người chơi đăng ký, đăng nhập vào hệ thống, thực hiện thao tác tham gia trò chơi.

1. **Phạm vi đề tài.**

Phạm vi nghiên cứu của đề tài:

* Đề tài xây dựng ứng dụng được sử dụng trong mạng LAN.
* Các client giao tiếp với server bằng công nghệ Java Socket.
* Ứng dụng đơn giản, hoàn thành được mục tiêu đề ra. Có thể giao tiếp tốt giữa client – server.

1. **Cấu trúc báo cáo.**

Báo cáo gồm các phần với các nội dung chính như sau:

* 1. Giới thiệu đề tài: Nội dung phần này sẽ tập trung giới thiệu tổng quan về đề tài, làm rõ mục tiêu, phạm vi đề tài và giới thiệu cấu trúc báo cáo.
* 2. Công nghệ ứng dụng, giải pháp, thuật toán, hiện thực hệ thống: Phần này là thân bài chính của bài báo cáo. Nội dung sẽ tập trung trình bày những kiến thức cần thiết cho quá trình thực hiện đề tài gồm những kiến thức tổng quan về Socket, port, Java Socket library, Thread, JSON. Bên cạnh đó đưa ra những phân tích cũng như cách thức hiện thực ứng dụng.
* 3 Kết quả: Nội dung của phần này là đưa ra kết quả cuối cùng của quá trình hiện thực. Hiển thị cách thức hoạt động, luồng chạy cơ bản của ứng dụng.
* 4. Kết luận: Nội dung của phần này là đánh giá quá trình hiện thực đề tài, nhìn nhận những mặt hạn chế và đưa ra những phương hướng phát triển, mở rộng đề tài và đưa vào thực tế.

1. **Công nghệ ứng dụng, giải pháp, thuật toán, hiện thực hệ thống**
2. ***Công nghệ ứng dụng***
   1. **Sơ lược về ứng dụng client – server**

Mô hình client-server là một mô hình quen thuộc trong mạng máy tính, được áp dụng rất rộng rã. Ý tưởng của mô hình này là máy con (client - đóng vài trò là máy khách) gửi một yêu cầu (request) để máy chủ (server - đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.

Việc giao tiếp giữa client và server được thực hiện dưới hình thức trao đổi các thông điệp (Message). Để được phục vụ, client sẽ gởi một thông điệp yêu cầu (Request Message) mô tả về công việc muốn server thực hiện. Khi nhận được thông điệp yêu cầu, server tiến hành phân tích để xác định công việc cần phải thực thi. Nếu việc thực hiện yêu cầu này có sinh ra kết quả trả về, server sẽ gởi nó cho client trong một thông điệp trả lời (Reply Message). Dạng thức (format) và ý nghĩa của các thông điệp trao đổi giữa client và server được qui định rõ bởi giao thức (protocol) của ứng dụng.

* 1. **Các khái niệm mạng máy tính cơ bản**

Để có thể thiết kế thành công ứng dụng mạng, trước hết chúng ta cần phải hiểu cách thức giao tiếp thông tin của chúng trong mạng máy tính.

Thông thường, khi một thiết bị muốn gửi thông tin cho thiết bị khác, thông tin đó sẽ được bao đóng qua các tầng khác nhau để đảm bảo mục đích bảo mật và tính toàn vẹn. Ở đây chúng ta sẽ tìm hiểu kiến trúc của mô hình mạng thông dụng nhất – mô hình TCP/IP.

* + 1. **Các loại giao thức mạng phổ biến**
       1. **Giao thức**

**Giao thức mạng:** Tập hợp tất cả các quy tắc, quy ước để đảm bảo cho các máy tính trên mạng có thể giao tiếp với nhau gọi là giao thức. Như vậy các máy trên mạng muốn giao tiếp với nhau thì phải có chung một giao thức.

Các giao thức mạng thường được những tổ chức có uy tín nghiên cứu, phát triển, công bố và được đại đa số các nhà sản xuất công nhận, áp dụng vào sản phẩm của mình. Thông thường các hầu hết các máy tính sử dụng các giao thức có thể giao tiếp với nhau mà không cần phải quan tâm đến chủng loại.

Một số giao thức phổ biến: TCP, IP, UDP, HTTP, FTP,…

**Giao thức ứng dụng:** Tương tự như giao thức mạng, giao thức ứng dụng là tập hợp các quy tắc, quy ước đảm bảo việc trao đổi thông tin của một ứng dụng giữa các máy tính trong một mạng được diễn ra dễ dàng và có tính hệ thống.

Thông thường các giao thức ứng dụng được nhà sản xuất ứng dụng quy định, và chỉ có tác dụng trong phạm vi của ứng dụng đó. Các giao thức này sẽ được sử dụng trước khi ứng dụng gửi dữ liệu và sau khi ứng dụng nhận dữ liệu nhằm phân tách các yêu cầu, thông tin được gửi từ ứng dụng của một máy tính khác trong mạng. Các giao thức ứng dụng sẽ liên quan chủ yếu đến các chức năng của ứng dụng đó.

Ứng dụng “Kéo – Búa – Bao” cũng cần có riêng cho mình một giao thức để có thể giao tiếp dễ dàng trong quá trình hoạt động.

* + - 1. **Giao thức TCP**

TCP (Transmission Control Protocol - "Giao thức điều khiển truyền vận") là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.

TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet và các ứng dụng kết quả, trong đó có WWW, thư điện tử và Secure Shell.

Trong bộ giao thức TCP/IP, TCP là tầng trung gian giữa giao thức IP bên dưới và một ứng dụng bên trên. Các ứng dụng thường cần các kết nối đáng tin cậy kiểu đường ống để liên lạc với nhau, trong khi đó, giao thức IP không cung cấp những dòng kiểu đó, mà chỉ cung cấp dịch vụ chuyển gói tin không đáng tin cậy.

* + - 1. **Giao thức UDP**

UDP (User Datagram Protocol) là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. Khác với TCP, UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.

Những ứng dụng phổ biến sử dụng UDP như DNS (Domain Name System), ứng dụng streaming media, Voice over IP, Trivial File Transfer Protocol (TFTP).

* 1. ***Socket, port và Java socket***

Kiến trúc OSI có thể phân thành hai tầng lớn là là Hosts layer gần với người dùng cuối hơn và Media layer gần với các thiết bị mạng hơn. Có thể xem Tầng giao vận (Transport) là tầng nằm giữa 2 phía, làm bản lề cho sự giao tiếp thông tin của ứng dụng mạng. Chúng ta sẽ tìm hiểu kỹ hơn về tầng này.

* + 1. **Socket**

**Socket** là một cổng logic mà một chương trình sử dụng để kết nối với một chương trình khác chạy trên một máy tính khác trên Internet. Chương trình mạng có thể sử dụng nhiều Socket cùng một lúc, nhờ đó nhiều chương trình có thể sử dụng Internet cùng một lúc.

Có 2 loại Socket:

+ **Stream Socket**: Dựa trên giao thức TCP (Tranmission Control Protocol) việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

**+ Datagram Socket**: Dựa trên giao thức UDP (User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 quá trình. Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thế không đúng trình tự và lặp lại. Tuy nhiên vì nó không yêu cầu thiết lập kết nối không phải có những cơ chế phức tạp nên tốc độ nhanh.

*Có thể nhận thấy Stream Socket sử dụng một cơ chế truyền thông tin đáng tin cậy, vì vậy, chúng ta sẽ tập trung nghiên cứu cách áp dụng loại hình socket này trong ứng dụng “Kéo búa bao”.*

* + 1. **Port**

**Port** xác định duy nhất một quá trình (process) trên một máy trong mạng. Hay nói cách khác là cách mà phân biệt giữa các ứng dụng.

VD: Khi máy bạn chạy nhiều ứng dụng mạng như Yahoo, Firefox, game online…

Ví dụ chương Yahoo sử dụng (port 5150 hay 5050) thì khi ai đó gửi tin nhắn đến cho bạn, lúc tin nhắn đến máy bạn nó sẽ dựa vào port để nhận biết đó là

chương trình Yahoo (port 5150) chứ ko pải là chương trình khác. Sau đó thông tin sẽ đc xử lý và hiễn thị tin nhắn lên.

– Một TCP/IP Socket gồm một địa chỉ IP kết hợp với một port xác định duy nhất một tiến trình (process) trên mạng. Hay nói cách khác Luồng thông tin trên mạng dựa vảo IP là để xác định máy một máy trên mạng còn port xác định 1 tiến trình trên 1 máy.

* + 1. **Java socket**

Nhằm tạo sự thuận tiện trong việc lập trình, Java cung cấp cho chúng ta các lớp các lớp Socket được sử dụng để biểu diễn kết nối giữa client và server. Gói java.net cung cấp hai lớp - Socket và ServerSocket - thể hiện kết nối giữa client và server.

Các Socket cung cấp kỹ thuật giao tiếp giữa hai máy tính sử dụng TCP. Một chương trình Client tạo một socket trên đầu cuối của giao tiếp và cố gắng để kết nối socket đó tới một Server.

Khi kết nối được tạo, Server tạo một đối tượng Socket trên đầu cuối của giao tiếp. Client và Server bây giờ có thể giao tiếp bằng việc đọc và ghi từ Socket.

Lớp java.net.Socket biểu diễn một Socket, và lớp java.net.ServerSocket cung cấp một kỹ thuật cho chương trình Server để nghe thông tin từ các Client và thành lập các kết nối với chúng.

**Các bước sau xảy ra khi thành lập một kết nối TCP giữa hai máy tính sử dụng Socket:**

Server khởi tạo một đối tượng ServerSocket, biểu thị số hiệu cổng (port) nào để xuất hiện giao tiếp.

Server gọi phương thức *accept()* của lớp ServerSocket. Phương thức này đợi tới khi một Client kết nối với Server trên cổng đã cho.

Sau khi Server đang đợi, một Client khởi tạo một đối tượng Socket, xác định tên Server và số hiệu cổng để kết nối tới.

Constructor của lớp Socket cố gắng để kết nối Client tới Server và số hiệu cổng đã xác định. Nếu giao tiếp được thành lập, bây giờ Client có một đối tượng Socket có khả năng giao tiếp với Server.

Trên Server-side, phương thức *accept()* trả về một tham chiếu tới một socket mới trên Server mà được kết nối với socket của Client.

Sau khi các kết nối được thành lập, giao tiếp có thể xảy ra bởi sử dụng I/O stream. Mỗi Socket có cả một *OutputStream* và *InputStream*. *OutputStream* của Client được kết nối với *InputStream* của Server, và *InputStream* của Client được kết nối với *OutputStream* của Server.

TCP là một giao thức giao tiếp hai chiều, vì thế dữ liệu có thể được gửi qua cả hai luồng tại cùng một thời điểm. Các lớp hữu ích sau đây cung cấp đầy đủ các phương thức để triển khai các Socket.

* 1. ***Đa luồng (multi-threading)***
     1. **Đa luồng là gì**

Đa luồng hay còn được gọi là Multithreading. Một chương trình đa luồng luôn có 2 tiến trình trở lên chạy song song nhau, mỗi tiến trình đó người ta gọi là một luồng (thread). Luồng là đơn vị nhỏ nhất trong java có thể thực hiện được 1 công việc riêng biệt và các luồng được quản lý bởi máy ảo java (Java Virtual Machine - JVM). Một ứng dụng java ngoài luồng chính có thể có các luồng khác thực thi đồng thời. Đa luồng giúp cho các tác vụ được xử lý độc lập giúp công việc được hoàn thành nhanh chóng. Vậy đa luồng có thể hiểu đơn giản là quá trình xử lý nhiều thread song song nhau và thực hiện các nhiệm vụ khác nhau cùng một lúc.

Server của ứng dụng cần phải quản lý rất nhiều người dùng truy cập đến một lúc, multithread là lựa chọn thích hợp để giải quyết vấn đề này.

* + 1. **Java Thread**

Ngôn ngữ java cung cấp cho ta đối tượng Thread nhằm thể hiện cơ chế đa luồng. Có hai cách chính để tạo luồng đó là tạo 1 đối tượng của lớp được thừa kế từ lớp Thread hoặc implements từ giao diện Runnable.

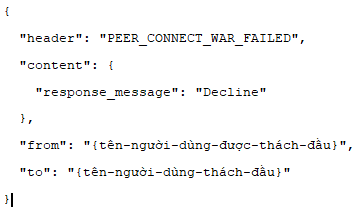
So sánh hai cách tạo ra thread hay dùng ở trên:

+ Giống nhau: cùng xử lý để có thể tạo ra một luồng

+ Khác nhau: nếu bạn tạo thread bằng cách implements từ interface Runnable thì bạn có thể kể thừa một class khác ngoài Class Thread, còn nếu bạn tạo thread bằng cách kế thừa Class Thread thì bạn sẽ không thể extend class nào khác vì Java không hỗ trợ đa kế thừa.

* 1. ***JSON***
     1. **Giới thiệu về JSON.**

JSON là chữ viết tắt của Javascript Object Notation, đây là một dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được, bạn có thể sử dụng lưu nó vào một file, một record trong CSDL rất dễ dàng. JSON có định dạng đơn giản, dễ dàng sử dụng và truy vấn hơn XML rất nhiều nên tính ứng dụng của nó hiện nay rất là phổ biến.



Hình 1 Cấu trúc cơ bản của một chuỗi JSON

**Cú pháp của một chuỗi JSON:**

Cú pháp của JSON rất đơn giản là mỗi thông tin dữ liệu sẽ có 2 phần đó là key và value, điều này tương ứng trong CSDL là tên field và giá trị của nó ở một record nào đó. Tuy nhiên nhìn qua thì đơn giản nhưng nếu ta mổ xẻ nó ra thì có một vài điều như sau:

Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}.

Các key, value của JSON bắt buộc phải đặt trong dấu nháy kép {"}, Nếu có nhiều dữ liệu (nhiều cặp key => value) thì ta dùng dấu phẩy (,) để ngăn cách.

Các key của JSON bạn nên đặt chữ cái không dấu hoặc số, dấu \_ và không có khoảng trắng. Ký tự đầu tiên không nên đặt là số.

* + 1. **Kiểu dữ liệu của JSON**

JSON có 5 kiểu dữ liệu chính

- Number: kiểu số bao gồm số nguyên và số thực.

- String: kiểu chuỗi, nội dung bao bởi cặp dấu nháy kép “, những ký tự đặt biệt được escape bởi dấu \.Theo chuẩnJSON thì không sử dụng dấu nháy đơn như Javascript để bọc chuỗi.

- Boolean: kiểu luận lý bao gồm 2 giá trị là true và false

- Array: kiểu mảng, gồm các phần tử phân cách nhau bởi dấu phẩy ‘,’ và mảng được bao bởi cặp dấu [].

- Object: kiểu đối tượng, gồm những cặp giá trị đi cùng nhau, mỗi cặp phân cách bởi dấu phẩy’,’, đối tượng được bao bởi cặp dấu { và }, cặp giá trị bao gồm tên và giá trị được phân cách bởi dấu hai chấm’:’.

- Null: giá trị null.

* + 1. **Ứng dụng của chuỗi JSON**

+ Lưu trữ dữ liệu đơn thuần:

- Cấu trúc của JSON gồm các key, value bên cạnh đó còn lưu được các kiểu dữ liệu cơ bản nên thường được sử dụng để lưu trữ các dữ liệu đơn giản dưới dạng chuỗi. Dữ liệu được lưu trữ bằng JSON thường là những dữ liệu đơn giản, gọn, nhẹ.

+ Chuẩn hóa, mô tả dữ liệu:

- Các dữ liệu có cấu trúc dễ dàng được chuyển đổi sang chuỗi JSON để truyền tải qua lại trong cùng một máy tính hay giữa các máy tình trong mạng, sau khi truyền tải thì chuỗi JSON cũng dễ dàng chuyển đổi ngược lại thành dữ liệu như cũ.

1. **Phân tích chức năng hệ thống**
   1. ***Mô tả kiến trúc phần mềm.***

Ứng dụng trò chơi “Kéo Búa Bao” sẽ là ứng dụng mạng theo mô hình client – server với cấu trúc như sau:

- Một Server với SocketServer được mở sẵn và thực hiện 2 chức năng chính sau:

+ Chờ đợi kết nối từ phía client.

+ Nhận – xử lý – phản hồi các yêu cầu từ phía client dưới dạng thông điệp (Message).

*Server phải đảm bảo có thể quản lý, xử lý, phản hồi được nhiều client cùng lúc.*

- Các Client với

+ Socket được tạo ra nhằm gửi các yêu cầu và nhận kết quả từ Server.

+ Màn hình trận đấu để quản lý việc thi đấu giữa hai Client với nhau.

- Một Database Server quản lý người dùng và trạng thái của họ (Đang offline/online/trong trận).

* 1. ***Chức năng chi tiết***
     1. ***Chức năng của server***

Server của ứng dụng gồm có các chức năng chính sau:

* Nhận kết nối và thông điệp của các client.
* Xử lý các yêu cầu của client và trả về kết quả, cụ thể:
  + Nếu user đăng ký thành viên thì kiểm tra thông tin trong danh sách user đã tồn tại:

Nếu thông tin user không tồn tại trước đó trong danh sách, trả về thông báo đăng ký thành công và thêm user mới và cơ sở dữ liệu.

Nếu thông tin user đã tồn tại trong hệ thống, trả về thông báo đăng ký không thành công và yêu cầu user sử dụng thông tin khác để đăng ký.

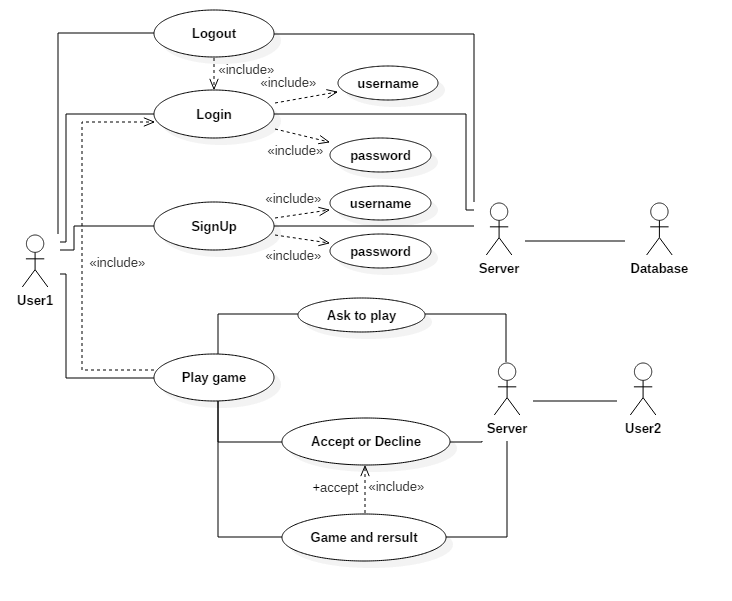
* + Nếu user đã đăng ký thành viên và đăng nhập thì kiểm tra trong database đã có user chưa, nếu có thay đổi trạng thái của user trong database thành online, thêm user vào danh sách người chơi và gửi danh sách cập nhật về cho các user hiện đang online.
  + Nếu có yêu cầu thi đấu giữa một người chơi với một người chơi khác, kiểm tra xem người được thách đấu có đang trong trận không, nếu không gửi lời mời đến người được thách đấu.
  + Nếu có người được thách đấu chấp nhận tham gia trận đấu, gửi thông tin chấp nhận về cho người thách đấu và bắt đầu trận đấu.

* Trận đấu sẽ đếm ngược trong 10 giây theo thời gian của Server. Sau 10 giây Server nhận kết quả lựa chọn gửi về từ client và tính toán người thắng, cuối cùng gửi thông tin của người chiến thắng về cho cả hai người chơi.

* + 1. ***Chức năng của client***

Clien là ứng dụng gồm các chức năng sau:

* Nhận kết nối đến server thông qua IP và port của nó.
* Cho phép người dùng nhập thông tin đăng ký, sau đó gửi thông tin này về cho server, thông báo với người dùng về tình trạng đăng ký có thành công hay không dựa trên thông điệp hồi đáp từ server.
* Cho phép người dùng nhập thông tin đăng nhập, sau đó gửi thông tin về cho server, nếu server chấp thuận đăng nhập, mở ra màn hình danh sách người chơi, nếu không, thông báo lỗi cho người dùng.
* Cho phép người dùng đăng xuất, sau khi đăng xuất thành công với server, các kết nối đến server sẽ được client hủy.
* Thay đổi thông tin danh sách người chơi dựa trên thông báo của server mỗi khi có người chơi đăng nhập hoặc đăng xuất.
* Cho phép người dùng thách đấu hoặc thông báo người dùng khác thách đấu, nếu lời thách đấu được chấp thuận, cửa sổ quản lý trận đấu sẽ được mở ra, trận đấu kéo dài trong vòng 10 giây, sau đó gửi lựa chọn về phía server.
* Thông báo người thắng cuộc dựa trên tính toán được gửi về từ server. Nếu có 1 trong 2 người không chọn vũ khí, người chọn vũ khí mặc định chiến thắng.
  + 1. ***Use Case Diagram***



Hình 2 Sơ đồ Use Case của ứng dụng

* + 1. ***Use Case Scenario***
* ***Use Case Login (người dùng đăng nhập)***

|  |  |
| --- | --- |
| Use case | Login |
| Actor | User |
| Main | Đăng nhập |
| Detail | Người dùng đăng nhập vào server bằng username và password. |

* ***Use Case Log out (Chức năng người dùng đăng xuất khỏi hệ thống)***

|  |  |
| --- | --- |
| Usecase | Log out |
| Actor | User |
| Main | Đăng xuất |
| Detail | Đăng xuất ra khỏi server khi người dùng không có nhu cầu muốn chơi nữa |
| Drerequisite | User phải đăng nhập thành công |

* ***Use Case Sign up (Chức năng thiết lập tài khoản người dùng)***

|  |  |
| --- | --- |
| Usecase | Sign Up |
| Actor | User |
| Main | Đăng ký |
| Detail | Thiết lập tài khoản người dùng dựa trên username và password nhập vào từ user. |

* ***Use Case Play game (Chức năng người chơi kết nối đế tham gia chiến đấu)***

|  |  |
| --- | --- |
| Usecase | Play game |
| Actor | User |
| Main | Chiến đấu |
| Detail | Cho phép người chơi thách đấu với người chơi khác.  Chấp nhập hoặc từ chối thách đấu.  Chiến đấu nếu lời thách đấu được chấp nhận |
| Drerequisite | Yêu cầu đăng nhập |

* + 1. ***Tổng quan về cấu trúc cơ sở dữ liệu của hệ thống.***

Hệ thống cơ sở dữ liệu của hệ thống chủ yếu được sử dụng để quản lý thông tin người dùng và các trạng thái của người dùng như online, offline, đang chơi. Cơ sở dữ liệu được liên kết trực tiếp với server và chỉ có thể được truy vấn từ server.

Hệ quản trị cơ sở dữ liệu được sử dụng là mysql.

* + 1. ***Mô hình cấu trúc cơ sở dữ liệu.***

Cấu trúc cơ sở dữ liệu gồm 2 bảng (Hình 3.1):

**Bảng user:** chịu trách nhiệm lưu thông tin người dùng

**+ id (Primary key):** số thứ tự

**+ username:** tên đăng nhập của người dung.

**+ password:** mật khẩu dưới dạng mã hóa md5.

**+ status (Foreig Key – Reference status.id):** trạng thái của người dùng, các trạng thái này được lưu trong bảng status.

**Bảng status:** chịu trách nhiệm lưu thông tin các trạng thái của người dùng

**+ id (Primary key):** số thứ tự

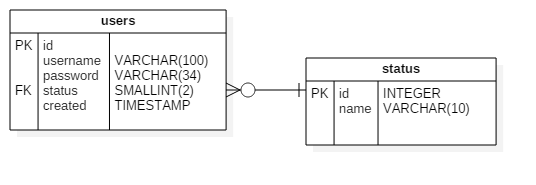
**+ name:** tên trạng thái (0: offline, 1:online, 2: trong trận)

Trạng thái dùng để xác định người dùng có được phép thực hiện một hành động nào đó không.

**- Ví dụ:**

**+** Nếu người dùng đăng nhập vào tài khoản với password đúng và trạng thái đang là 0: offline 🡪 đăng nhập hợp lệ.

**+** Nếu người dùng đăng nhập vào tài khoản với password đúng và trạng thái của tài khoản là 1: online hoặc 2: trong trận 🡪 đăng nhập không hợp lệ.



Hình 3 Mô hình cấu trúc cơ sở dữ liệu

1. **Hiện thực hệ thống.**
   1. ***Giao thức của ứng dụng***

Để ứng dụng có thể trao đổi dữ liệu, trao đổi yêu cầu một cách thống nhất và đảm bảo các thông điệp được gói gọn trong một số hành động nhất định cần có một giao thức ứng dụng chung.

Giao thức nên được xây dựng dưới dạng chuỗi và theo dạng cấu trúc của JSON. Sau đây là chi tiết các giao thức.

* + 1. ***Cấu trúc cơ bản của một thông điệp***

Một thông điệp sẽ gồm có 4 phần:

**+ header:** Phần đầu của thông điệp, giúp phân biệt loại yêu cầu hoặc loại kết quả trả về. Các header thường không thay đổi và được sử dụng ở cả client và server nên được lưu dưới dạng biến số hằng (constant).

**+ content:** Nội dung của thông điệp, phần này chứa các nội dung đi kèm, giúp cụ thể hóa thông điệp, cung cấp những thông tin quan trọng để phía nhận có thể xử lý chính xác.

**+ from:** Nơi gởi thông điệp.

**+ to:** Nơi nhận thông điệp.

Giả sử client muốn gửi yêu cầu đăng nhập đến server, thông điệp sẽ có dạng như sau:

{

"header": "HELLO",

"content": {

"username": "trungtm",

"password": "e10adc3949ba59abbe56e057f20f883e"

},

"from": "trungtm",

"to": "SERVER"

}

* + 1. ***Giao thức các thông điệp gửi từ phía client***

- **Yêu cầu đăng ký:**

{

"header": "SIGNUP",

"content": {

"username": "{tên-người-dùng-đăng-ký}",

"password": "{mật-khẩu-người-dùng-đăng-ký}"

},

"from": "{tên-người-dùng-đăng-ký}",

"to": "SERVER"

}

- **Yêu cầu đăng nhập:**

{

"header": "HELLO",

"content": {

"username": "{tên-người-dùng-đăng-nhập}",

"password": "{mật-khẩu-người-dùng-đăng-nhập}"

},

"from": "{tên-người-dùng-đăng-nhập}",

"to": "SERVER"

}

- **Yêu cầu đăng xuất:**

{

"header": "BYE",

"content": {

"username": "{tên-người-dùng-đăng-xuất}",

"password": "{mật-khẩu-người-dùng-đăng-xuất}" },

"from": "{tên-người-dùng-đăng-xuất}",

"to": "SERVER"

}

- **Yêu cầu thách đấu với đối thủ khác:**

{

"header": "CLIENT\_CONNECT\_WAR",

"content": {

"username": "{tên-người-dùng-được-thách-đấu}"

},

"from": "{tên-người-dùng-được-thách-đấu}",

"to": "SERVER"

}

- **Chấp nhận yêu cầu thách đấu:**

{

"header": "PEER\_CONNECT\_WAR\_SUCCESS",

"content": {

"response\_message": "Accept"

},

"from": "{tên-người-dùng-được-thách-đấu}",

"to": "{tên-người-dùng-thách-đấu}"

}

- **Từ chối yêu cầu thách đấu:**

{

"header": "PEER\_CONNECT\_WAR\_FAILED",

"content": {

"response\_message": "Decline"

},

"from": "{tên-người-dùng-được-thách-đấu}",

"to": "{tên-người-dùng-thách-đấu}"

}

- **Báo thoát trận đấu:**

{

"header": "PEER\_GAME\_DISCONNECT",

"content": {

"response\_message": "Disconnect"

},

"from": "{người-chơi-thoát}",

"to": "{người-chơi-còn-lại}"

}

\* Khi trận đấu đang diễn ra và có một người chơi thoát trận, người còn lại sẽ chiến thắng.

- **Lựa chọn vũ khí sau cùng:**

{

"header": "SHOW\_CHOICE",

"content": {

"choice": "{Búa/Kéo/Bao}"

},

"from": "trungtm1",

"to": "SERVER"

}

* + 1. ***Giao thức các thông điệp gửi từ phía server***

- **Chấp nhận yêu cầu đăng ký:**

{

"header": "SIGNUP\_SUCCESS",

"content": {

"response\_content": "Đăng ký thành công."

},

"from": "SERVER",

"to": "{tên-người-dùng-đăng-ký}"

}

- **Báo lỗi yêu cầu đăng ký:**

{

"header": "SIGNUP",

"content": {

"response\_content": "{thông-điêp-lỗi}"

},

"from": "SERVER",

"to": "{tên-người-dùng-đăng-ký}"

}

- **Chấp nhận yêu cầu đăng nhập:**

{

"header": "WELCOME",

"content": {},

"from": "SERVER",

"to": "{tên-người-dùng-đăng-nhập}"

}

- **Báo lỗi yêu cầu đăng nhập:**

{

"header": "HELLO",

"content": {

"response\_content": "{thông-điệp-lỗi}"

},

"from": "SERVER",

"to": "{tên-người-dùng-đăng-nhập}"

}

- **Thông báo người dùng khác vừa đăng nhập:**

{

"header": "NEWCOMER",

"content": {

"{tên-người-dùng-1}": "{trạng-thái:online|trong trận}"

"{tên-người-dùng-2}": "{trạng-thái:online|trong trận}"

…

"{tên-người-dùng-n}": "{trạng-thái:online|trong trận}"

},

"from": "SERVER",

"to": "{một-người-dùng-đang-online}"

}

\* thông điệp này được gửi cho tất cả người dùng hiện đang đăng nhập

\* content: chứa tên tất cả người dùng hiện đang đăng nhập và trạng thái

- **Chấp nhận** y**êu cầu đăng xuất:**

{

"header": "LOGOUT\_SUCCESS",

"content": {},

"from": "SERVER",

"to": "{tên-người-dùng-đăng-xuất}"

}

- **Thông báo người dùng khác vừa đăng xuất:**

{

"header": "NEWOUTER",

"content": {

"{tên-người-dùng-1}": "{trạng-thái:online|trong trận}"

"{tên-người-dùng-2}": "{trạng-thái:online|trong trận}"

…

"{tên-người-dùng-n}": "{trạng-thái:online|trong trận}"

},

"from": "SERVER",

"to": "{một-người-dùng-đang-online}"

}

\* thông điệp này được gửi cho tất cả người dùng hiện đang đăng nhập

\* content: chứa tên tất cả người dùng hiện đang đăng nhập và trạng thái

- **Chấp nhận yêu cầu thách đấu với đối thủ khác:**

{

"header": "CONNECT\_WAR\_SUCCESS",

"content": {

"response\_content": "waiting",

"otherUsername": "{tên-người-chơi-được-thách-đấu}"

},

"from": "SERVER",

"to": "{tên-người-chơi-thách-đấu}"

}

\* Thông điệp được tạo ra khi có yêu cầu thách đấu và SERVER kiểm tra thấy người chơi được thách đấu hiện không trong trận.

- **Báo lỗi yêu cầu thách đấu thách đấu với đối thủ khác:**

{

"header": "CONNECT\_WAR\_FAILED",

"content": {

"response\_content": "Đối thủ hiện đang trong trận."

},

"from": "SERVER",

"to": "{tên-người-dùng-thách-đấu}"

}

\* Thông điệp được tạo ra khi có yêu cầu thách đấu và SERVER kiểm tra thấy người chơi được thách đấu hiện đang trong trận.

- **Trung gian** y**êu cầu thách đấu từ người thách đấu đến người được thách đấu:**

{

"header": "PEER\_CONNECT\_WAR",

"content": {

"otherUsername": ""{tên-người-dùng-thách-đấu}"

},

"from": "{tên-người-dùng-thách-đấu}",

"to": ""{tên-người-dùng-được-thách-đấu}"

}

- **Trung gian báo hiệu người được thách đấu chấp nhận trận đấu:**

{

"header": "PEER\_CONNECT\_WAR\_SUCCESS",

"content": {

"response\_message": "Accept"

},

"from": "{tên-người-dùng-được-thách-đấu}",

"to": "{tên-người-dùng-thách-đấu}"

}

- **Trung gian báo hiệu người được thách đấu từ chối trận đấu:**

{

"header": "PEER\_CONNECT\_WAR\_FAILED",

"content": {

"response\_message": "Decline"

},

"from": "{tên-người-dùng-được-thách-đấu}",

"to": "{tên-người-dùng-thách-đấu}"}

}

- **Đếm ngược trong trận đấu:**

{

"header": "GAME\_COUNT\_DOWN",

"content": {

"count\_down": {10 -> 0}

},

"from": "SERVER",

"to": "{tên-người-chơi}"

}

\* thông điệp này được gửi cho tất cả người dùng hiện đang cùng chơi một trận.

- **Yêu cầu người chơi gửi lựa chọn vũ khí sau cùng:**

{

"header": "SHOW\_CHOICE",

"content": {

"message": "show\_choice"

},

"from": "SERVER",

"to": "{tên-người-chơi}"

}

\* thông điệp này được gửi cho tất cả người dùng hiện đang cùng chơi một trận.

- **Kết quả trận đấu:**

{

"header": "SHOW\_RESULT",

"content": {

"result": "{tên-người-thắng}",

"trungtm1": "Búa",

"trungtm": "Kéo"

},

"from": "SERVER",

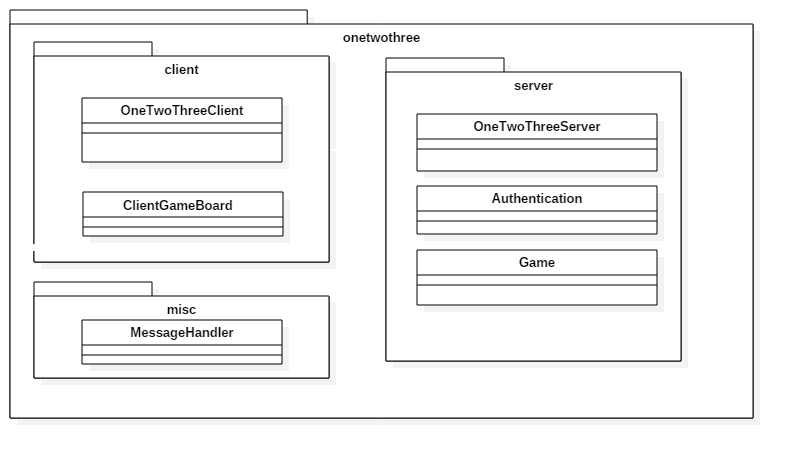
"to": "{tên-người-chơi}"}

\* thông điệp này được gửi cho tất cả người dùng hiện đang cùng chơi một trận.

* 1. ***Sơ đồ class của ứng dụng***
     1. ***Sơ đồ class tổng quát***

Sơ đồ class của ứng dụng được chia làm ba package: client, server và phần dùng chung mics:

+ **Package client:** các class trong package này chịu trách nhiệm giúp cho phía client hoạt động, bao gồm giao diện màn hình đăng nhập, giao diện màn hình chơi game, các hàm chức năng giúp cho việt kết nối và giao tiếp với server.

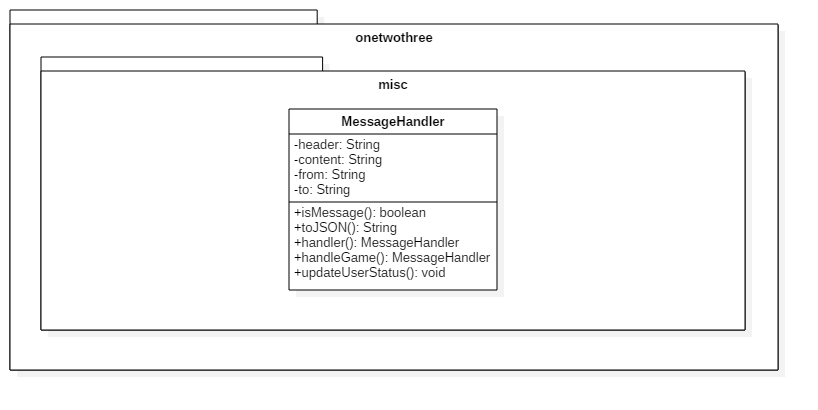


Hình 4 Cấu trúc class tổng quan của ứng dụng

+ **Package server:** các class trong package này chịu trách nhiệm giúp cho phía server hoạt động, bao gồm một màn hình lớn hiển thị lịch sử người dùng và danh sách người dùng đang online, các hàm chức năng giúp cho việt kết nối và giao tiếp với server.

+ **Package mics:** class MessageHandle trong package này chịu trách nhiệm hỗ trợ cả phía client và server. Là class đại diện cho thông điệp của giao thức ứng dụng với các thuộc tính có cấu trúc như cấu trúc của giao thức ứng dụng. Chịu trách nhiệm phân tích chuỗi ký tự thành dạng JSON, phân tích và xử lý các thông điệp và trả về kết quả cho client hoặc server để xử lý tiếp tục.

* + 1. ***Sơ đồ class cho phần dùng chung mics***

******

Hình 5 Sơ đồ class của package onetwothree.misc

* **Class MessageHandle**

Server và client sẽ liên lạc với nhau bằng giao thức ứng dụng đã đề cập phần trên, các thông điệp này ở dạng JSON, MessageHandle có trách nhiệm chuyển các chuỗi ký tự thông điệp JSON thành dạng Object dễ quản lý. Bên cạnh đó class này còn có nhiệm vụ phân tích các thông điệp, xử lý thông điệp và trả về thông điệp kết quả để client hoặc server gửi đi hồi đáp.

* ***Mô tả các thuộc tính***

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Nhiệm vụ** |
| header | String | Lưu trữ phần **header** của một thông điệp trong ứng dụng. |
| content | String | Lưu trữ phần **content** của một thông điệp trong ứng dụng. |
| from | String | Lưu trữ phần **from** của một thông điệp trong ứng dụng. |
| to | String | Lưu trữ phần **to** của một thông điệp trong ứng dụng. |

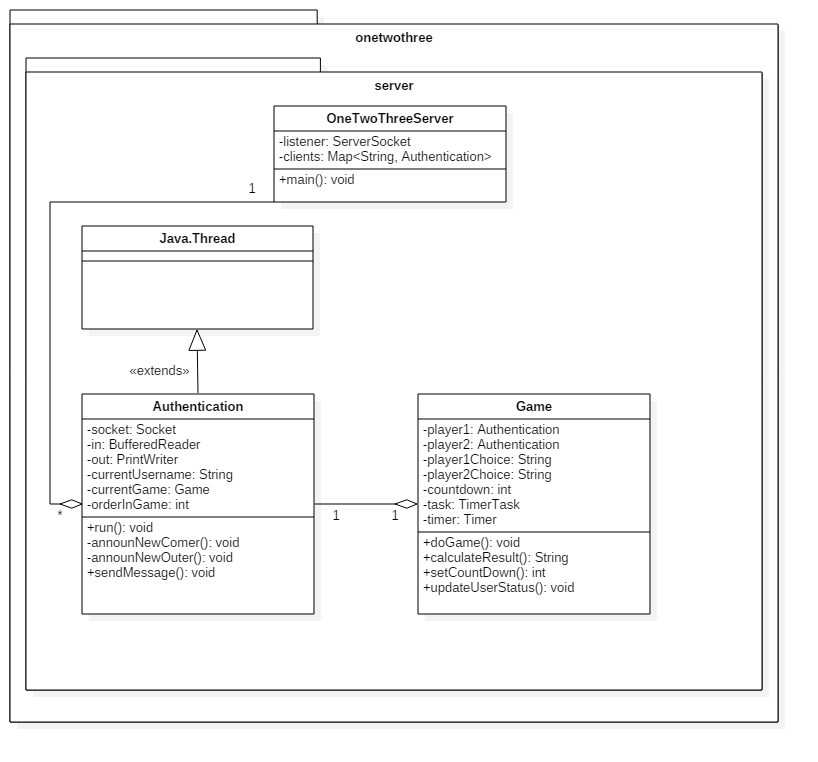
Bảng 1 - Mô tả thuộc tính class MessageHandler

* ***Mô tả các phương thức***

|  |  |  |
| --- | --- | --- |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Nhiệm vụ** |
| isMessage() | boolean | Kiểm tra một chuỗi thông điệp có đúng cấu trúc của giao thức đã đặt ra không.  Nếu đúng trả về **true.**  Nếu sai trả về **false.** |
| toJSON() | String | Chuyển đổi các thuộc tính **header, content, from, to** sang dạng chuỗi có cấu trúc JSON. |
| handler() | MessageHandler | Nhận một thông điệp từ server, hàm phân tích thông điệp là yêu cầu gì từ phía client cho server (thông qua **header**), có nội dung như thế nào (thông qua **content**) sau đó xử lý và trả về thông điệp kết quả cho server. |
| handleGame() | MessageHandler | Tương tự như handler() nhưng những thông điệp mà handleGame xử lý chỉ liên quan đến những sự kiện xảy ra trong khi hai người chơi đang giao đấu |
| updateUserStatus() | void | **Parameter:**  + *username* (String): tên người dùng  + *status* (int): trạng thái  Thay đổi trạng thái của user có tên là *username* sang trạng thái *status* trên database |

Bảng 2 - Mô tả các phương thức class MessageHandler

* + 1. ***Sơ đồ class cho phía server***

******

Hình 6 Sơ đồ class của package onetwothree.server

* **Class OneTwoThreeServer**

Là class chính được khởi tạo khi server được mở. Có chức năng tạo ra ServerSocket chờ client kết nối, tạo ra các đối tượng để quản lý người dùng.

* ***Mô tả các thuộc tính***

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Nhiệm vụ** |
| listener | ServerSocket | ServerSocket được tạo trên port 8901, có nhiệm vụ lắng nghe các kết nối từ phía client, từ đó tạo một đường truyền riêng để quản lý việc trao đổi thông điệp giữ từng client với server |
| clients | Map<String, Authentication> | Quản lý thông tin của các người dùng đang online. |

Bảng 3 - Mô tả thuộc tính class OneTwoThreeServer

* ***Mô tả các phương thức***

|  |  |  |
| --- | --- | --- |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Nhiệm vụ** |
| main() | void | Đây là hàm chính của ứng dụng ở phía server, các nhiệm vụ của hàm này bao gồm:  + Tạo ServerSocket *listener*  + Mỗi khi có kết nối từ client, tạo ra một đường truyền, một đối tượng Authentication để quản lý thông tin user và thông tin trao đổi qua lại giữa server và client mới. |

Bảng 4 - Mô tả phương thức class OneTwoThreeServer

* **Class Authentication**

Class Authentication được tạo ra khi có một client kết nối với server, mỗi object của class đại diện cho một client. Nhiệm vụ chính của class này là nhận, phân tích – xử lý, trao đổi thông tin, thông điệp với một client kết nối đến server. Class Authentication extends class Thread để có thể xử lý đa luồng, giúp class OneTwoThreeServer có thể quản lý trao đổi thông tin với nhiều client cùng lúc.

* ***Mô tả các thuộc tính***

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Nhiệm vụ** |
| socket | Socket | Tạo ra một đường truyền riêng thông qua một socket mới giữa server và client vừa kết nối. |
| in | BufferedReader | Quản lý các thông điệp được gửi từ client |
| out | PrintWritter | Quản lý các thông điệp được gửi từ server |
| currentUsername | String | Biến này dùng để lưu trữ tên người dùng nếu người dùng yêu cầu đăng nhập và đăng nhập thành công. |
| currentGame | Game | Tạo ra một object kiểu Game để lưu trữ thông tin giao chiến hiện tại của người dùng và một người dùng khác |
| orderInGame | int | Lưu trữ thứ tự của người chơi trong trận đấu (1: đại diện cho người thách đấu, 2: đại diện cho người được thách đấu) |

Bảng 5 - Mô tả thuộc tính class Authentication

* ***Mô tả các phương thức***

|  |  |  |
| --- | --- | --- |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Nhiệm vụ** |
| run() | void | Đây là hàm chính của ứng dụng, chạy xuyên suốt để lắng nghe và trao đổi thông điệp với client.  Ứng với mỗi thông điệp hàm sẽ tạo ra một object của class MessageHandler và gọi hàm handler để xứ lý và chờ kết quả thông điệp, sau đó gửi thông điệp kết quả về cho client |
| announNewComer() | void | Gửi một thông điệp đến tất cả client hiện đang online để báo với các client rằng có một user vừa đăng nhập thành công, nội dung thông điệp là danh sách tất cả user hiện đang online |
| announNewOuter() | void | Gửi một thông điệp đến tất cả client hiện đang online để báo với các client rằng có một user vừa đăng xuất, nội dung thông điệp là danh sách tất cả user hiện đang online |
| sendMessage() | void | Hàm này được sử dụng để gửi một thông điệp là object của class MessageHandler, hàm thường được gọi từ phía ngoài class, nơi mà các object khác không có thông tin về biến *out.* |

Bảng 6 - Mô tả phương thức class Authentication

* **Class Game**

Class Game được tạo ra khi có hai client bắt đầu tham gia vào một trận đấu. Nhiệm vụ chính của class này là nhận, phân tích – xử lý, trao đổi thông tin, thông điệp với cả hai client tham gia. Phân tích xử lý để đưa ra kết quả người thắng cuộc sau cùng.

* ***Mô tả các thuộc tính***

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Nhiệm vụ** |
| player1 | Authentication | Một object của class Authentication, giúp quản lý người chơi thách đấu. |
| player2 | Authentication | Một object của class Authentication, giúp quản lý người chơi được thách đấu. |
| player1Choice | String | Thông tin lựa chọn của người chơi 1, giá trị mặc định là chuỗi rỗng. |
| Player2Choice | String | Thông tin lựa chọn của người chơi 2, giá trị mặc định là chuỗi rỗng. |
| countdown | int | Thời gian đếm ngược của trận đấu, giá trị ban đầu là 10, giảm sau mỗi giây. |
| task | TimerTask | Tạo ra một TimerTask có nhiệm vụ thực hiện việc gửi thông tin về thời gian đếm ngược, sau mỗi giây. |
| timer | Timer | Quản lý của biến *task,*  có nhiệm vụ bắt đầu *task* mỗi khi trận đấu bắt đầu, sau khi thời gian đếm ngược kết thúc, dừng *task,* kích hoạt hàm tính toán người thắng. |

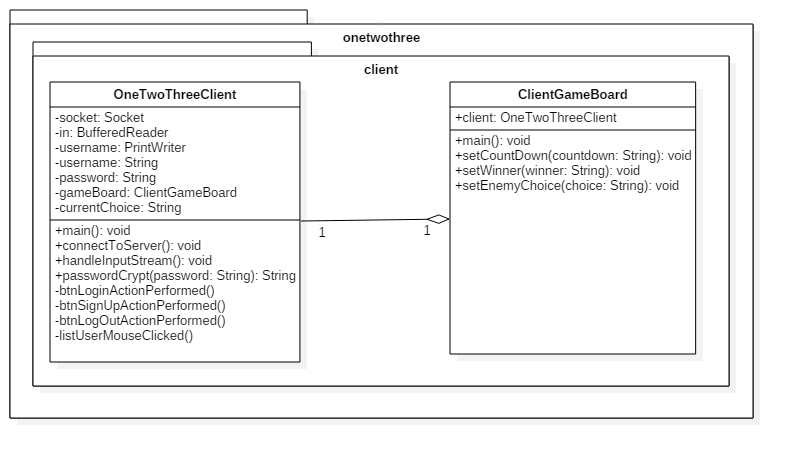
Bảng 7 - Mô tả các thuộc tính class Game

* ***Mô tả các phương thức***

|  |  |  |
| --- | --- | --- |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Nhiệm vụ** |
| doGame() | void | Phương thức được thực hiện khi trận đấu bắt đầu, *timer* kích hoạt *task* để task gửi các thông điệp về đếm ngược. |
| setCountDown() | int | Trả về giá trị hiện tại của biến *countdown -* ứng với thời gian đếm ngược, khi thời gian đếm ngược là 0, gọi lệnh dừng *task*, gửi thông điệp yêu cầu client gửi lựa chọn của người chơi, đồng thời kích hoạt hàm calculateResult() |
| calculateResult() | String | Tính toán kết quả dựa trên lựa chọn của user mà client trả về. Theo quy luật:   1. Hai lựa chọn bằng nhau sẽ hòa. 2. Trong hai lựa chọn nếu chỉ có một lựa chọn là chuỗi rỗng thì người lựa chọn còn lại thắng 3. Búa thắng Kéo, Kéo thắng Bao, Bao thắng Búa. |
| updateUserStatus() | void | Thay đổi *status* của người chơi về lại 1: online sau khi cả hai đã kết thúc màn chơi. |

Bảng 8 - Mô tả phương thức class Game

* + 1. ***Sơ đồ class phía Client***

******

Hình 7 Sơ đồ class của package onetwothree.client

* **Class OneTwoThreeClient**

Là class chính được khởi tạo khi client được mở. Có chức năng tạo ra Socket để kết nối đến server cũng như trao đổi thông điệp. Tạo ra đối tượng quản lý việc thi đấu giữa user và một user khác.

* ***Mô tả các thuộc tính***

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Nhiệm vụ** |
| socket | Socket | Tạo ra một socket và gửi yêu cầu kết nối về phía server, nếu kết nối thành công, đây là biến quản lý việc trao đổi thông điệp giữa client với server |
| in | BufferedReader | Quản lý các thông điệp được gửi từ server |
| out | PrintWritter | Quản lý các thông điệp được gửi từ client |
| username | String | Biến này dùng để lưu trữ tên người dùng nếu người dùng yêu cầu đăng nhập và đăng nhập thành công. |
| password | String | Lưu trữ password của người dùng dưới dạng mã hóa md5 |
| gameBoard | ClientGameBoard | Đối tượng của class ClientGameBoard, được khởi tạo khi trận đấu được chấp thuận bởi 2 người dùng. Có chức năng hiển thị nội dung trận đấu. |
| currentChoice | String | Lựa chọn hiện tại của người dùng, trong quá trình trận đấu, người dùng có thể liên tục thay đổi lựa chọn của mình, nó sẽ được lưu vào biến *currentChoice,* cho đến khi thời gian đếm ngược kết thúc và có yêu cầu gửi lựa chọn từ phía server.  Giá trị mặc định của biến này là null, khi giá trị là null, nghĩa là người dùng không chọn, client không cần gửi thông điệp về lựa chọn. |

Bảng 9 - Mô tả thuộc tính class OneTwoThreeClient

* ***Mô tả các phương thức***

|  |  |  |
| --- | --- | --- |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Nhiệm vụ** |
| main() | void | Có nhiệm vụ kích hoạt tạo kết nối đến server, khởi tạo màn hình giao diện, quản lý thông điệp vào ra. |
| connectToServer() | void | Tạo ra một socket kết nối đến ServerSocket port 8901. |
| handleInputStream() | void | Quản lý các thông điệp gửi về từ phía server, đa phần là các thông điệp kết quả cho những yêu cầu trước đó, dựa trên nội dung kết quả, thay đổi giao diện cho phù hợp. |
| passwordCrypt(String password) | String | Mã hóa thông tin password nhập vào từ người dùng để đảm bảo tính bảo mật bằng md5. |
| btnLoginActionPerformed() | void | Thực hiện gửi thông điệp login khi có sự kiện click vào nút login |
| btnSignUpActionPerformed | void | Thực hiện gửi thông điệp signup khi có sự kiện click vào nút Sign Up |
| btnLogOutActionPerformed | void | Thực hiện gửi thông điệp logout khi có sự kiện click vào nút Đăng Xuất |
| listUserMouseClicked | void | Thực hiện gửi thông điệp thách đấu khi có sự kiện double-click vào tên một người dùng trong danh sách |

Bảng 10 - Mô tả phương thức class OneTwoThreeClient

* **Class ClientGameBoard**

Class này liên quan đến giao diện trận đấu, các thao tác xảy ra trong quá trình trận đấu. Được OneTwoThreeClient tạo ra khi bắt đầu trận đấu giữa user với một user khác.

* ***Mô tả các thuộc tính***

|  |  |  |
| --- | --- | --- |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Nhiệm vụ** |
| client | OneTwoThreeClient | Biến dùng để tham chiếu ngược về OneTwoThreeClient đã tạo ra nó. |

Bảng 11 - Mộ tả thuôc tính class ClientGameBoard

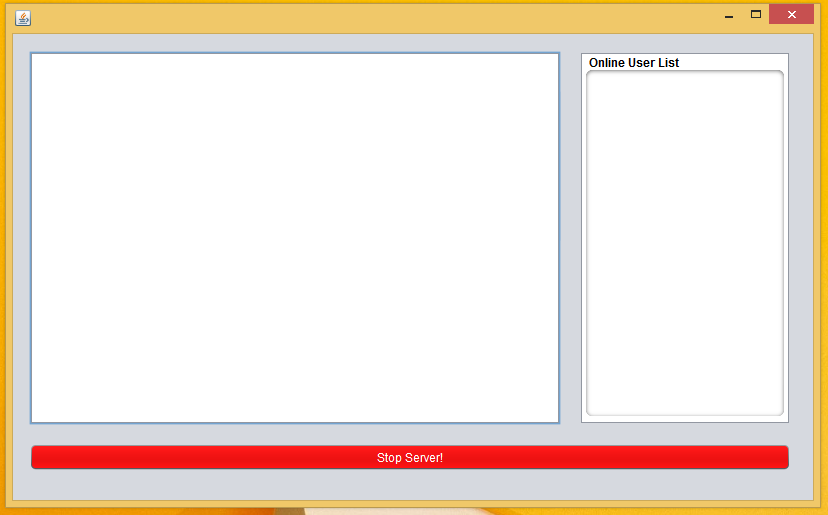
* ***Mô tả các phương thức***

|  |  |  |
| --- | --- | --- |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Nhiệm vụ** |
| main() | void | Phương thức để kích hoạt giao diện. |
| setCountDown(String countdown) | void | Hiển thị thời gian đếm ngược từ parameter countdown. Dựa trên thông điệp về đếm ngược từ server, OneTwoThreeClient phân tích thời gian còn lại và kích hoạt phương thức này. |
| setWinner(String winner) | void | Hiển thị người thắng cuộc cuối cùng từ parameter winner. Dựa trên thông điệp về kết quả từ server, OneTwoThreeClient phân tích người chiến thắng và kích hoạt phương thức này. |
| setEnemyChoice (String choice) | void | Hiển thị lựa chọn của đối thử lên giao diện. Dựa trên thông điệp về kết quả từ server, OneTwoThreeClient phân tích lựa chọn của đối thủ và kích hoạt phương thức này. |
| btnKeoActionPerformed () | void | Thực hiện thay đổi OneTwoThreeClient.currentChoice thành ‘Keo’ khi có sự kiện click vào nút Kéo |
| btnBuaActionPerformed | void | Thực hiện thay đổi OneTwoThreeClient.currentChoice thành ‘Bua’ khi có sự kiện click vào nút Búa |
| btnBaoActionPerformed | void | Thực hiện thay đổi OneTwoThreeClient.currentChoice thành ‘Bao’ khi có sự kiện click vào nút Bao |

Bảng 12 - Mô tả phương thức class ClientGameBoard

1. **Kết quả**
2. **Giao diện ứng dụng.**
   1. ***Server.***

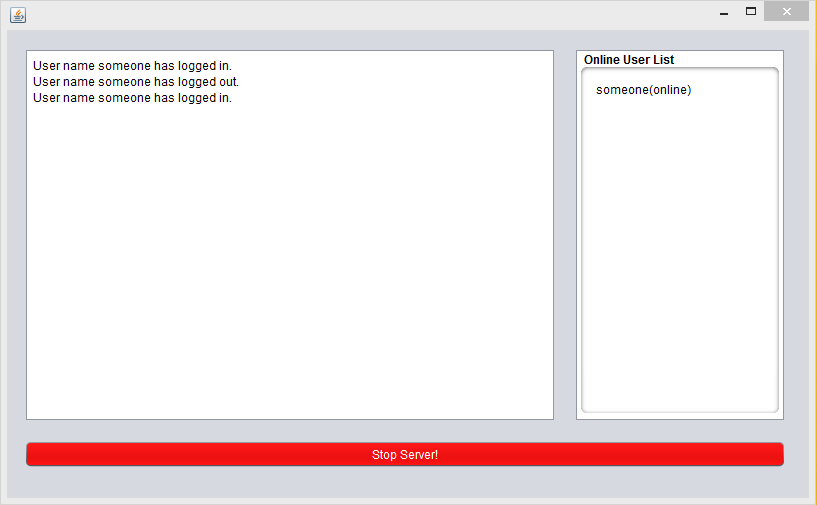
Ban đầu khi server được bật lên, giao diện sẽ bao gồm nhật ký đăng nhập, đăng xuất của người dùng ở bên trái, và danh sách người dùng đã đăng nhập vào hệ thống, cả hai vùng sẽ không có thông tin (Hình 3-1)



Hình 8 Giao diện bắt đầu của server

Khi có user đăng nhập thành công, server sẽ ghi nhận điều đó vào nhật ký, đồng thời ghi tên của user đó vào danh sách user hiện đang trong hệ thống.

Khi có user đăng xuất, server ghi nhận vào nhật ký, đồng thời loại tên của user đó ra khỏi danh sách user hiện trong hệ thống.



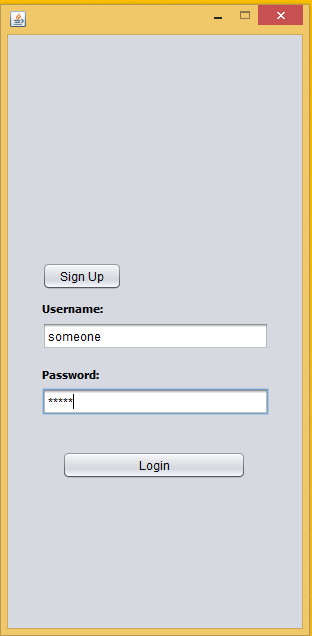
Hình 9 Giao diện nhật ký của server khi có người dùng đăng nhập / đăng xuất

* 1. ***Client.***
     1. ***Màn hình chính***

**Giao diện đăng nhập**

Giao diện đăng nhập gồm có hai trường username và password, nút Login dùng để kích hoạt các phương thức liên quan đến đăng nhập, nut Sign Up dùng để mở ra giao diện đăng ký.

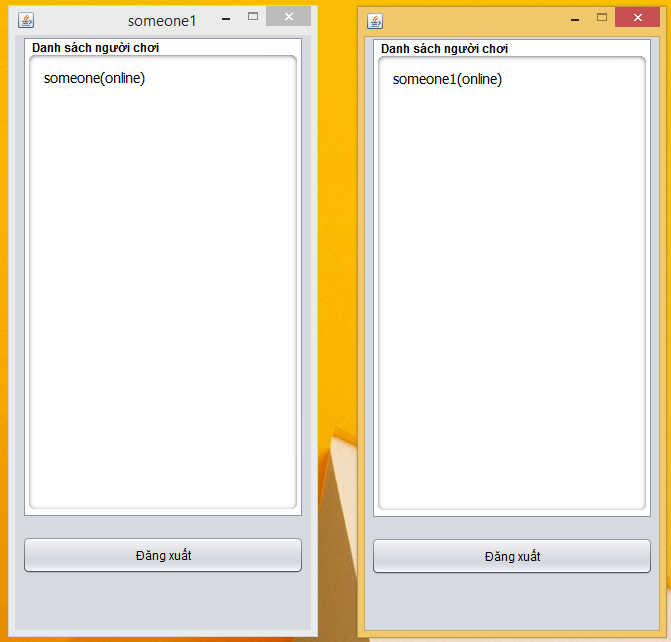
Khi đăng nhập thành công, giao diện sẽ hiển thị danh sách người chơi đang trong hệ thống và nút đăng xuất.

****

Hình 10 Giao diện đăng nhập thành công của client

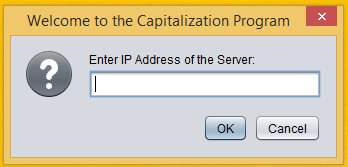
Hình 11 Giao diện đăng nhập của client

Sau khi người dùng đăng nhập thành công, một thông báo có người dùng mới vừa đăng nhập được gửi đến tất cả các người dùng đang trong hệ thống. Các client sau khi nhận được thông điệp này sẽ cập nhật lại danh sách người chơi.

****

Hình 12 Giao diện client sau khi có người dùng mới đăng nhập

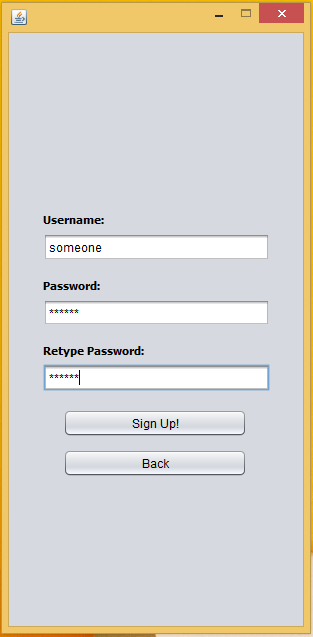
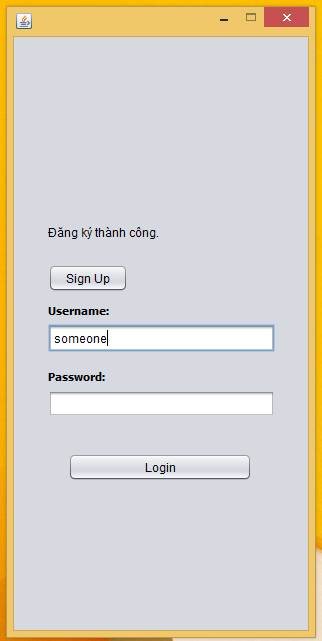
**Lựa chọn IP server**

Các thao tác đăng nhập và đăng xuất cần biết được IP của server trong LAN để có thể tạo kết nối đến

Hình 13 Bảng hỏi địa chỉ IP của server

**Giao diện đăng ký**

Giao diện đăng ký gồm có ba trường chính là username, password và retype password (nhập lại password). Được mở ra bằng cách ấn vào nút Sign Up ở màn hình đăng nhập

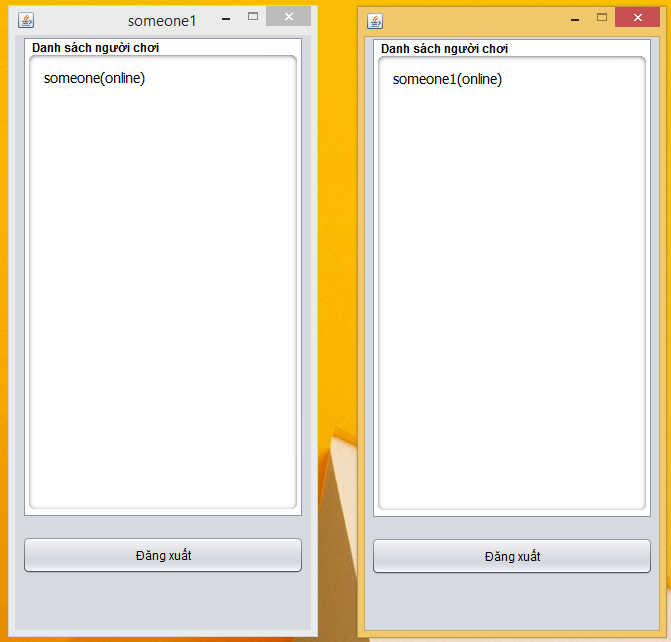
****

Hình 14 Giao diện đăng thành công của client

Hình 15 Giao diện đăng ký của client

**Giao diện đăng xuất**

Để đăng xuất, người dùng cấn ấn vào nút Đăng xuất nằm dưới danh sách người chơi.

****

Hình 16 Giao diện đăng xuất của client

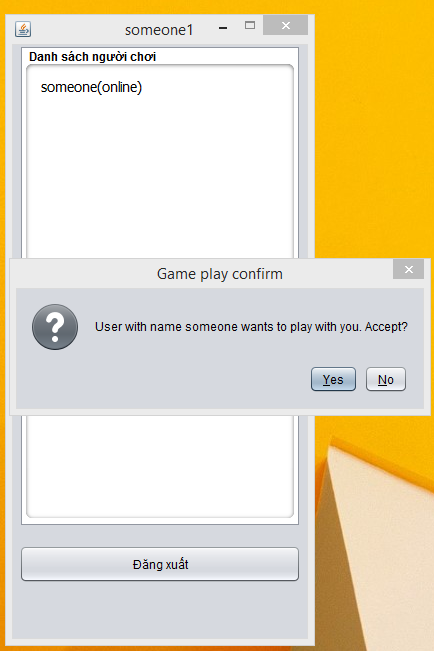
Sau khi ấn vào nút đăng xuất, người dùng sẽ đăng xuất khỏi hệ thống, trạng thái (status) sẽ được chuyển về 0, đồng thời một thông báo có người dùng mới vừa đăng xuất được gửi đến tất cả các người dùng đang trong hệ thống. Các client sau khi nhận được thông điệp này sẽ cập nhật lại danh sách người chơi.

* + 1. ***Màn hình trò chơi***

**Thách đấu**

Để có thể thách đấu với người dùng khác, người dùng nhấp đúp chuột vào tên người chơi đó. Sau khi nhận được thông điệp thách đấu, server sẽ kiểm tra trạng thái của người được thách đấu có phải đang là 1: online không. Nếu đúng, server sẽ gửi thông điệp thách đấu này sang cho người chơi được thách đấu.

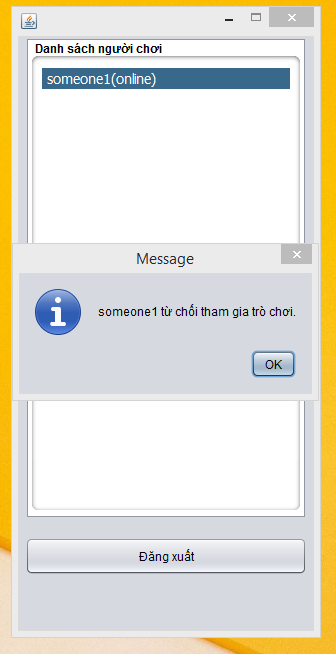
Tại phía người chơi được thách đấu, khi nhận được thông điệp thách đấu, client sẽ hiển thị một thông báo



Hình 17 Giao diện thông báo có người thách đấu

**Từ chối thách đấu**

Nếu người dùng nhận được thông điệp thách đấu và chọn No (từ chối), một thông điệp sẽ được gửi về người thách đấu



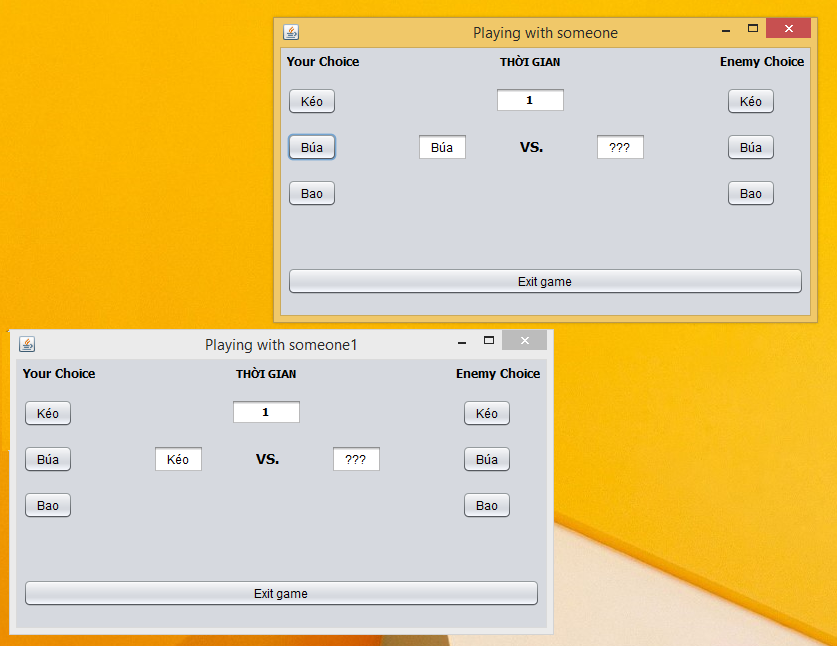
Hình 18 Giao diện thông báo từ chối thách đấu

**Chấp nhận thách đấu**

Nếu người dùng nhận được thông điệp thách đấu và chọn Yes (chấp nhận), một thông điệp sẽ được gửi về người thách đấu và ở cả hai phỉa người chơi, giao diện trò chơi sẽ được bật lên

**Giao diện trò chơi**

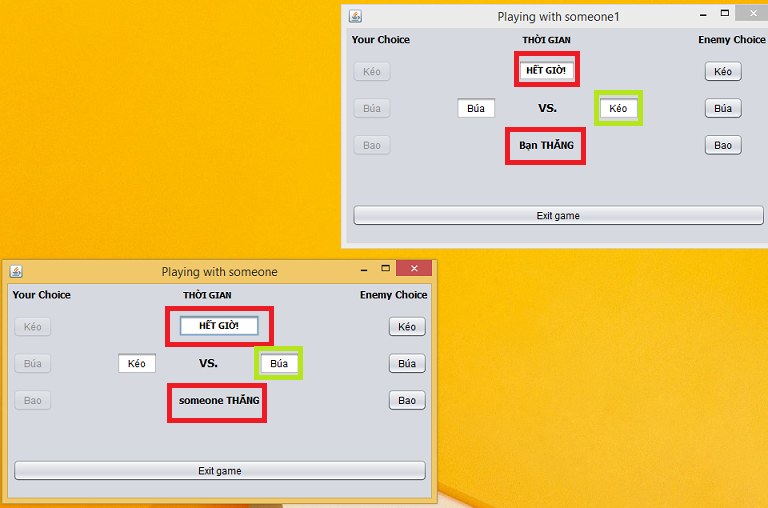
Tại giao diện trò chơi, thời gian sẽ được đếm ngược trong 10 giây, người chơi phải chọn vũ khí trước khi đếm ngược kết thúc. Trong thời gian đó người chơi cũng không thể biết đối phương lựa chọn vũ khí gì.



Hình 19 Giao diện trò chơi

Sau khi thời gian kết thúc, lựa chọn cuối cùng sẽ được gửi về phía server, ở đây server sẽ tính toán người tháng cuộc và trả kết quả về cho hai người chơi.

Ở phần kết quả, chúng ta sẽ biết được người chiến thắng, và lựa chọn của đối thủ.



Hình 20 Giao diện khi trò chơi kết thúc

1. **Đánh giá hệ thống.**
   1. ***Ưu điểm.***

* Hoàn chỉnh các chức năng cơ bản, đạt được các mục tiêu ban đầu đề ra.
* Xây dựng được hệ thống với giao diện thân thiện, dễ sử dụng.
* Ứng dụng chạy tốt với nhiều client cùng lúc, quản lý tốt trạng thái người dùng.
  1. ***Nhược điểm.***
* Giao diện còn chưa bắt mắt. Không thay đổi được lối chơi
* Lối chơi còn quá đơn giản.
* Không có các dịch vụ tích hợp.

1. **Kết luận**
2. **Kết luận.**

Về cơ bản, nhóm đã hoàn thành được mục tiêu ban đầu đặt ra là xây dựng ứng dụng trò chơi “Kéo Búa Bao” online với giao diện thân thiện, dễ dùng, bao gồm 2 thành phần là server và ứng dụng client.

Tuy nhiên, hệ thống của nhóm vẫn còn một số nhược điểm như đã trình bày ở phần trước. Nhưng, hệ thống cũng đã đáp ứng được những tính năng cơ bản nhất:

+ Tạo được một trò chơi giữa hai người chơi.

+ Về phía server:

* Quản lý được nhiều người chơi cùng lúc với đa luồng (multi-threading).
* Hiểu được về giao tiếp qua mạng với Socket, port và protocol.
* Kết nối với cơ sở dữ liệu, dùng cơ sở dữ liệu để quản lý người chơi và trạng thái người chơi, giúp cho việc quản lý được dễ dàng hơn.

+ Về phía client:

* Thay đổi giao diện tùy theo mục đích sử dụng của người dùng (đăng nhập, đăng xuất, đăng ký, chơi trò chơi).

Ứng dụng hiện tại có thể áp dụng thành trò chơi đơn giản trong mạng LAN. Để có thể áp dụng rộng rãi cần có nhiều cải tiến, phát triển.

1. **Hướng phát triển.**

Để ứng dụng có thể hoạt động hiệu quả hơn cần phải

* Cải thiện ứng dụng.
* Thay đổi giao diện đẹp hơn, hiện đại hơn
* Thay đổi lối chơi:
  + Thêm các vũ khí mới.
  + Thêm số màn chơi
  + Hệ thống tính điểm.
  + Hệ thống quá khứ đối đầu.
  + Thêm tính năng chat giữa người chơi với nhau
  + Thêm tính năng giúp người khác xem trận đấu
  + Thêm tính năng bạn bè.
* Phát triển mô hình bớt phụ thuộc vào mô hình client-server:
  + Chỉ dùng server để đăng nhập, đăng xuất, đăng ký và truy xuất thông tin người chơi khác
  + Khi thách đấu và tiến hành giao chiến có thể sử dụng mô hình P2P (peer – to – peer ) tạo đường kết nối trực tiếp giữa 2 người chơi.