TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN PHƯỚC SANG - 52100303
LÊ KHẮC THANH TÙNG - 52100943
NGUYỄN THANH TÚ - 52100349

WEB SOCKET SERVER DÙNG EXPRESSJS

BÁO CÁO GIỮA KỲ
PHÁT TRIỂN
ỨNG DỤNG WEB VỚI NODEJS

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

TRẦN PHƯỚC SANG - 52100303
LÊ KHẮC THANH TÙNG - 52100943
NGUYỄN THANH TÚ - 52100349

WEB SOCKET SERVER DÙNG EXPRESSJS

BÁO CÁO GIỮA KỲ
PHÁT TRIỂN
ỨNG DỤNG WEB VỚI NODEJS

Người hướng dẫn
GV. Nguyễn Thanh Quân

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn đến toàn thể giảng viên trường Đại học Tôn Đức Thắng nói chung và giảng viên GV. Nguyễn Thanh Quân nói riêng vì đã tạo môi trường học tập, phát triển toàn diện cho chúng em. Ngoài ra, trong suốt quá trình học tập chúng em đã được dạy và tiếp thu nhiều kiến thứ bổ ích, quan trọng đối với tương lai của chúng em.

Bằng kiến thức mà chúng em tiếp thu được trong suốt quá trình học tập tại trường để hoàn thiện bài báo cáo này. Tuy còn nhiều thiếu sót cần phải học hỏi và cải thiện trong bài báo cáo nên chúng em mong rằng sẽ được thầy nhận xét, đánh giá và cho ý kiến về bài báo cáo.

Chúng em rất mong có được sự góp ý đến từ thầy cô tại trường để biết được những thiếu sót của bản thân đồng thời cải thiện kỹ năng của chúng em. Cuối cùng, chúng em xin gửi lời cám ơn chân thành đến quý thầy cô trường Đại học Tôn Đức Thắng và thầy Nguyễn Thanh Quân.

TP. Hồ Chí Minh, ngày 1 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Trần Phước Sang

Lê Khắc Thanh Tùng

Nguyễn Thanh Tú

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của GV. Nguyễn Thanh Quân. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 1 tháng 12 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Trần Phước Sang

Lê Khắc Thanh Tùng

Nguyễn Thanh Tú

WEB SOCKET SERVER DÙNG EXPRESSJS

TÓM TẮT

Phần 1: Mở đầu và tổng quan đề tài

Phần 2: Tìm hiểu về WebSocket

Phần 3: Xây dựng một WebSocket cho phép người dùng chat trực tuyết sử dụng ExpressJS và Socket.io

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

Hbs	Handlebars

CHƯƠNG 1. MỞ ĐẦU VÀ TỔNG QUAN ĐỀ TÀI

1. Lý do chọn đề tài

Người dùng hiện nay rất ưa chuộng các trang web mang lại trải nghiệm sử dụng tốt. Một trong những tiêu chí để đánh giá trang web có trải nghiệm tốt có thể kể đến là việc người dùng cảm thấy dữ liệu được cập nhật theo thời gian thực.

Nếu dữ liệu không được cập nhật theo thời gian thực thì người dùng phải tải lại trang để có thể truy vấn lại từ trong cơ sở dữ liệu rồi hiển thị lên giao diện. Như vậy thì người dùng phải reload lại trang liên tục để có thể thấy được sự cập nhật dữ liệu.

Công nghệ khắc phục hạn chế trên chính là Websocket

2. Mục tiêu thực hiện đề tài

Tìm hiểu về giao thức WebSocket và xây dựng một trang web cho phép người dùng chat trực tuyến.

CHƯƠNG 2. WEB SOCKET PROTOCOL

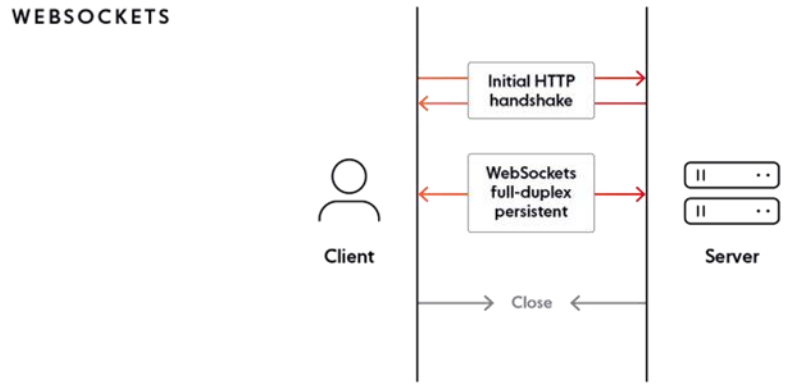
2.1 Tổng quan về WebSocket

WebSocket là một giao thức truyền tin dựa trên kết nối TCP.

WebSocket cho phép các kênh giao tiếp song song hai chiều giữa client và server bằng cách sử dụng một TCP socket để tạo một kết nối hiệu quả và ít tốn kém.

Kết nối giữa máy chủ và máy khách sẽ tồn tại cho đến khi một trong hai bên chấm dứt kết nối. Sau khi đóng kết nối bởi một trong hai máy, kết nối sẽ bị chấm dứt cả hai đầu.

Sử dụng WebSocket giúp giảm độ trễ và tăng hiệu suất so với việc sử dụng polling (kiểm tra liên tục) hoặc long polling (kết nối được mở trong một khoảng thời gian dài). WebSocket đã trở thành một phần quan trọng của các ứng dụng web hiện đại và được hỗ trợ rộng rãi trong nhiều ngôn ngữ lập trình và framework.



Hình 2.1: Tổng quan về WebSocket

2.2 Ưu và nhược điểm của WebSocket

2.2.1 Ưu điểm

- Cung cấp giao tiếp hai chiều mạnh mẽ: WebSocket cho phép client và server truyền dữ liệu cho nhau theo hai chiều, không cần phải thực hiện các request HTTP. (giao tiếp hai chiều có nghĩa là trước đây ở mô hình client-server thì client sẽ gửi request lên server rồi server response về dữ liệu tuy nhiên với WebSocket client hay server đều có thể gửi dữ liệu đến và response dữ liệu cho bên còn lại)
- Độ trễ thấp: WebSocket sử dụng giao thức TCP, có độ trễ thấp hơn HTTP giúp cho dữ liệu được truyền nhanh và chính xác.
- Tỉ lệ xảy ra lỗi thấp: WebSocket được cung cấp các API đơn giản và dễ sử dụng cho nhà phát triển.
- Phù hợp cho ứng dụng web thời gian thực: WebSocket phù hợp cho các ứng dụng web thời gian thực như trò chuyện trực tuyến, trò chơi trực tuyến, cập nhật dữ liệu thời gian thực và các ứng dụng tương tác người dùng khác.
- Không cần phải gửi request mới để nhận dữ liệu mới: Khi sử dụng HTTP, trình duyệt phải gửi yêu cầu mới để nhận dữ liệu mới từ máy chủ vì sau khi client request và server response về dữ liệu thì kết nối nó sẽ bị đóng và muốn mở kết nối lại thì phải gửi một request khác lên server. Tuy nhiên, với WebSocket, hai bên client và server có thể gửi dữ liệu cho nhau mà không cần phải yêu cầu mới.

2.2.2 Nhược điểm

- Tốn nhiều tài nguyên: WebSocket sử dụng kết nối mở liên tục giữa trình duyệt và máy chủ, vì vậy nó sẽ tốn nhiều tài nguyên hơn so với HTTP.
- Không được hỗ trợ bởi một số trình duyệt cũ: Một số phiên bản cũ của các trình duyệt không hỗ trợ WebSocket.

2.3 Ứng dụng

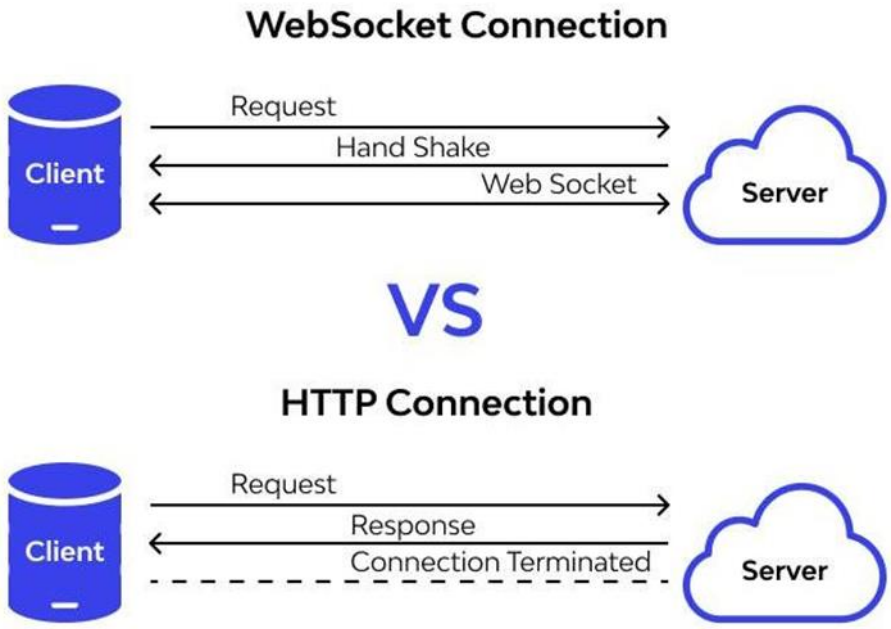
- Truyền dữ liệu:
    - Cập nhật tỷ số thể thao: Các trang web thể thao sử dụng WebSocket để cung cấp cập nhật thời gian thực về tỷ số trận đấu, thống kê và bình luận theo từng lượt chơi.
- Trò chuyện trực tiếp:
    - Hỗ trợ khách hàng qua trò chuyện: Các hệ thống hỗ trợ khách hàng trực tuyến sử dụng WebSocket để tạo điều kiện tương tác thời gian thực giữa nhân viên và khách hàng, cho phép phản hồi ngay lập tức cho các truy vấn và khắc phục sự cố.
    - Trò chuyện trên mạng xã hội: Các nền tảng mạng xã hội và phòng chat sử dụng WebSocket để cho phép các cuộc trò chuyện thời gian thực giữa người dùng, hỗ trợ trò chuyện nhóm, tin nhắn riêng và thông báo.
- Đồng bộ dữ liệu:
    - Công cụ chỉnh sửa cộng tác: Các công cụ chỉnh sửa cộng tác dựa trên đám mây, chẳng hạn như Google Docs sử dụng WebSocket để đồng bộ hóa các thay đổi được thực hiện bởi nhiều người dùng trong thời gian thực.
- Theo dõi vị trí thời gian thực:
    - Theo dõi GPS và quản lý đội xe: Các công ty vận tải và hậu cần sử dụng WebSocket để theo dõi vị trí thực tế của phương tiện của họ,

cung cấp cập nhật thời gian thực về giao hàng, tối ưu hóa tuyến đường và trạng thái tài xế.

- Ứng dụng chia sẻ vị trí cá nhân: Các ứng dụng di động được thiết kế để chia sẻ vị trí cá nhân sử dụng WebSocket để chia sẻ thông tin vị trí thời gian thực với bạn bè và gia đình.

- Thông báo và cảnh báo trong ứng dụng:
  - Thông báo tin tức mới: Các trang web tin tức và ứng dụng di động sử dụng WebSocket để cung cấp thông báo tin tức mới cho người dùng dựa trên các chủ đề và sở thích đã chọn của họ.
  - Thông báo mạng xã hội: Các nền tảng mạng xã hội sử dụng WebSocket để gửi thông báo thời gian thực về tin nhắn mới, lời mời kết bạn, . . .

2.4 Phân biệt giữa WebSocket và HTTP



Hình 2.4: Sự khác nhau giữa WebSocket và HTTP

WebSocket	HTTP
Full Duplex	Half Duplex
Stateful Protocol	Stateless Protocol
Truyền dữ liệu liên tục	Truyền dữ liệu không liên tục
Không đóng kết nối mỗi khi thực hiện xong một Request – Response	Chỉ mở kết nối trong một chu trình Request – Response trả về cho client và nó sẽ đóng kết nối sau khi làm xong

Bảng 2.4: Sự khác nhau giữa WebSocket và HTTP

2.5 Cơ chế hoạt động

Giao thức bao gồm 2 phần:

- Handshake:
    - Từ phía Client:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

    - Từ phía Server:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

- Data transfer: Trong giao thức Socket dữ liệu được truyền bằng cách sử dụng các chuỗi frame

2.6 WebSocket API

Đây là API được cung cấp sẵn trên browser.

Bao gồm các thành phần cơ bản như:

- Hàm khởi tạo một WebSocket: new WebSocket(url, protocols)
- Phương thức:
  - Send: Khi sự kết nối được thiết lập, người dùng sẽ sẵn sàng để nhận (gửi) thông điệp từ (đến) WebSocket server.
  - Close: Kết nối WebSocket được đóng thông qua phương thức close(). Sau khi phương thức close() được gọi sẽ không còn bất cứ dữ liệu nào được trao đổi.
- Thuộc tính:
  - binaryType: loại dữ liệu nhị phân sẽ được nhận thông qua kết nối WebSocket. Giá trị của thuộc tính dạng String “blob” (mặc định) hoặc “arraybuffer”.
  - bufferedAmount: trả về số byte của dữ liệu được đưa vào hàng đợi. Giá trị thuộc tính sẽ được gán về 0 nếu tất cả dữ liệu bên trong hàng đợi đã được gửi đi.
  - extensions: một chuỗi trống hoặc danh sách các extensions được quy định bởi kết nối.
  - protocol: chuỗi hoặc danh sách các chuỗi tên giao thức con của server.
  - readyState: thuộc tính trạng thái hiện tại của kết nối WebSocket bao gồm các giá trị tương ứng: 0 (CONNECTING), 1 (OPEN), 2 (CLOSING), 3 (CLOSED)
  - url: đường dẫn URL tuyệt đối để kết nối của WebSocket do hàm khởi tạo quy định.
- Sự kiện: Open, Message, Error, Close.

CHƯƠNG 3. WEB SOCKET SERVER VỚI EXPRESSJS

3.1 Socket.IO

3.1.1 Giới thiệu

- Socket.IO là một thư viện JavaScript được sử dụng để triển khai WebSocket trên nền tảng web. Nó cung cấp API dễ sử dụng và mạnh mẽ để xây dựng ứng dụng thời gian thực (real-time).
- Socket.IO không chỉ hỗ trợ WebSocket mà còn có khả năng tự động chuyển đổi sang HTTP long-polling khi WebSocket không khả dụng. Do đó ứng dụng vẫn có thể hoạt động bình thường và liên tục kể cả khi các trình duyệt không hỗ trợ giao thức WebSocket.
- Điểm nổi bật của thư viện Socket.IO như:
    - Hỗ trợ nhiều trình duyệt và nền tảng, cho phép viết ứng dụng đa nền tảng dễ dàng.
    - Hỗ trợ các sự kiện tùy chỉnh và gửi dữ liệu phức tạp giữa máy chủ và khách hàng.
    - Socket.IO có khả năng mở rộng dễ dàng bằng cách phân phối kết nối và gửi sự kiện đến tất cả các client kết nối với nhiều server.
- Link: https://socket.io/

3.1.2 API

3.1.2.1 Client API

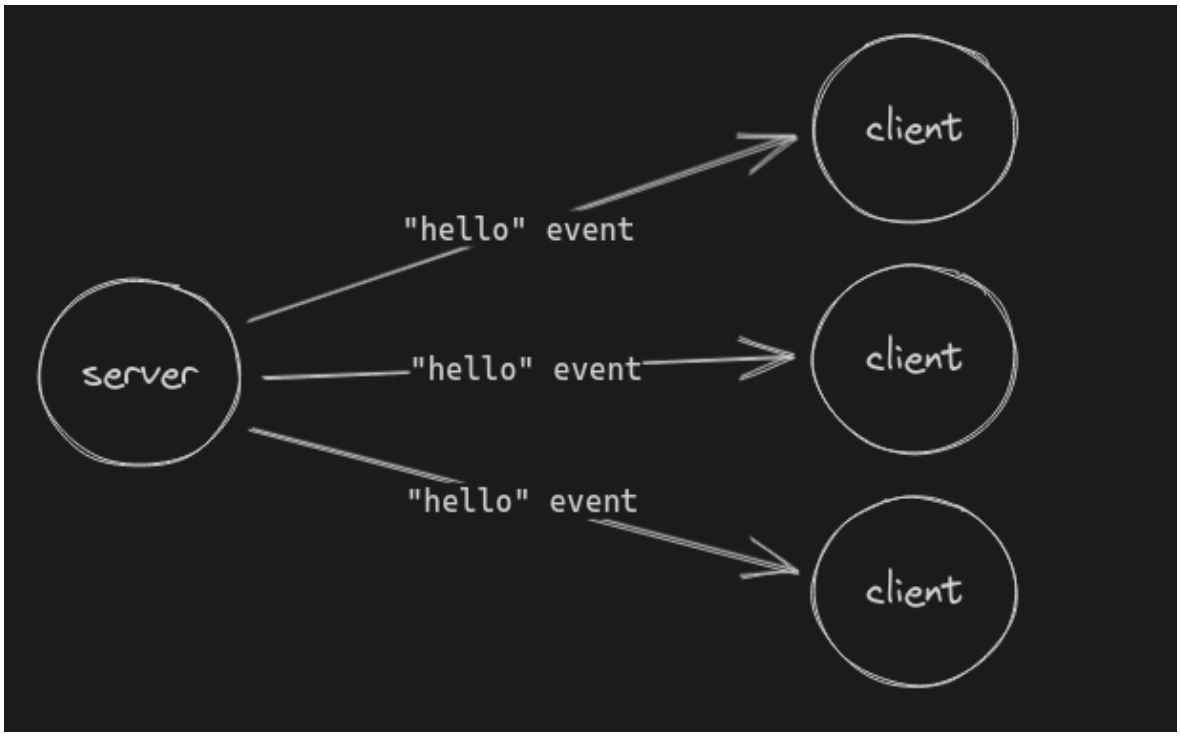
- **IO**
    - io([url][, options])
        - url <string> (mặc định window.location.host)
        - options <Object>
        - Trả về <Socket> để gửi hoặc nhận event từ server

- **Socket**
    - Thuộc tính
        - id: mã định danh duy nhất cho socket session. Được thiết lập sau khi sự kiện kết nối được kích hoạt và được cập nhật sau sự kiện kết nối lại
    - Phương thức
        - socket.emit(eventName[, ...args][, ack]) Phát ra sự kiện từ socket
            - eventName <string> | <symbol>
            - args <any[]>
            - ack <Function>
            - Trả về true
        - **socket.on(eventName, callback)** Đăng ký hàm xử lý cho sự kiện tương ứng
            - eventName <string> | <symbol>
            - listener <Function>
            - Returns <Socket>

3.1.2.2 Server API

- Server
    - Phương thức khởi tạo
        - new Server(httpServer[, options])
            - httpServer <http.Server> | <https.Server>
            - options <Object>
    - Các sự kiện
        - connection: kích hoạt khi có kết nối từ client
            - socket (Socket) client socket
        - io.emit(eventName[, ...args]): Phát ra sự kiện cho tất cả client đang được kết nối trong namespace chính

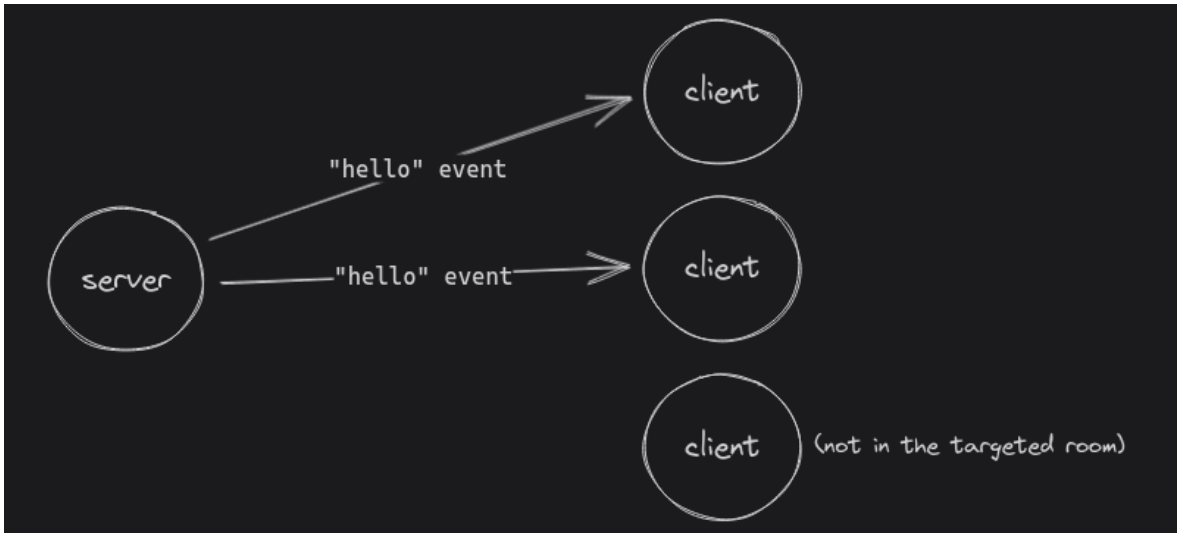
- eventName <string> | <symbol>
- args any[]
- Trả về true



Hình 3.1.2.2.1: Mô tả sự kiện io.emit

Nguồn: https://socket.io/docs/v4/tutorial/api-overview

- io.on(eventName, listener) Thêm hàm xử lý sự kiện cho sự kiện được đặt tên theo biến eventName vào cuối danh sách các hàm xử lý sự kiện
- io.to(room) Việc phát các sự kiện sẽ được gửi broadcast chỉ cho các client đã tham gia một room nào đó.
    - room <string> | <string[]>
    - Returns BroadcastOperator for chaining



Hình 3.1.2.2.2: Mô tả sự kiện io.to(room)

Nguồn: https://socket.io/docs/v4/tutorial/api-overview

- Socket
    - Các sự kiện
        - disconnect (Kích hoạt khi ngắt kết nối)
            - reason <string> lý do của việc ngắt kết nối (cho cả client và server)
    - Các phương thức
        - socket.join(room) Thêm socket vào một hoặc nhiều room
            - room <string> | <string[]>
            - Returns void | Promise
        - socket.leave(room) Xóa socket khỏi room.
            - room <string>
            - Returns void | Promise
        - socket.on(eventName, callback) Đăng ký hàm xử lý sự kiện cho sự kiện tương ứng
            - eventName <string> | <symbol>
            - callback <Function>
            - Returns <Socket>

3.2 Xây dựng ứng dụng chat

3.2.1 Cấu trúc thư mục

Ứng dụng được triển khai theo mô hình MVC:

Hình 3.2.1: Cấu trúc thư mục mô hình MVC

- config: cấu hình cho express app (kết nối cơ sở dữ liệu,..)
- controllers: chứa các hàm tiếp nhận request, xử lý rồi trả về response
- models: xây dựng các Schema, Model cho cơ sở dữ liệu
- public: chứa các tài nguyên tĩnh như css, js, image, video,...
- routers: định nghĩa các HTTP method, path với các hàm xử lý tương ứng (các hàm xử lý có thể từ controller)
- socket: chứa các hàm xử lý sự kiện bên phía server
- views: giao diện người dùng

3.2.2 Xây dựng server WebSocket

- Code xây dựng server WebSocket được đặt ở file app.js và các hàm xử lý sự kiện cho các sự kiện gửi lên từ socket client được đặt trong socket/chat.js
- Khai báo các thư viện cần thiết cho một server WebSocket như: express, socket.io, http, . . .

```
const http = require("http");

const express = require("express");

const { Server } = require("socket.io");
```

- Ngoài ra hệ thống cần sử dụng các thư viện đi kèm khác để thực hiện các chức năng xác thực, kết nối cơ sở dữ liệu như jsonwebtoken, dotenv, mongoose, . .
- Khởi tạo một server:

```
const app = express();

const server = http.createServer(app);

const io = new Server(server);
```

- Đăng ký các hàm xử lý sự kiện cho các sự kiện được gửi từ client socket

```
io.on("connection", (socket) => {

  console.log("user " + socket.id + " connected");


  socket.on("logout", (token) => { logout(io, token) })


  socket.on("login", (token) => { login(io, token) })


  socket.on("disconnect", () => { console.log("user
disconnected"); });


  socket.on("create-conversation", (token, receiverID)
=> { createConversation(io, token, receiverID) })


  socket.on("join-room", (roomID, token) => {
joinRoom(socket, roomID, token) });


  socket.on("send-message-room", (conversationID,
message, token, socketID) => { sendMessage(io,
conversationID, message, token, socketID) });
```

```
  socket.on("typing", (conversationID, socketID, token)
=> { typingMessage(io, conversationID, socketID, token)
});


  socket.on("send-file-room", (conversationID, file,
token, socketID, fileName) => { sendFile(io,
conversationID, file, token, socketID, fileName) });


})
```

3.2.3 Xây dựng client socket

- Trong thư mục views, 2 file xử lý socket là **index.handlebars** và **login.handlebars**
- Khởi tạo đối tượng Socket

```
  const socket = io();
```

- Đăng ký các hàm xử lý sự kiện cho các sự kiện được phát từ server socket. Các hàm xử lý sự kiện này được đặt trong thư mục **public/js/chat.js**

```
// SEND MESSAGE
  form.addEventListener('submit', (e) => {
sendMessage(e) });

  // RECEIVE MESSAGE
  socket.on('message', (msg, socketID, img,
conversationID) => { receiveMessage(msg, socketID, img,
conversationID) });

  socket.on('response-create-conversation', (data) => {
responseCreateConversation(data) })

  // Notification On-Off
  socket.on("notifi-online", (token) => {
notifiOnline(token) })
  socket.on("notifi-offline", (token) => {
notifiOffline(token) })

  $(document).ready(() => {
getAllUserAndRenderInSideBar() });

  $('#btn-create-group').click(() => {
createConversationMany() })

  $(document).ready(() => { getAllConversationOfUser()
})
```

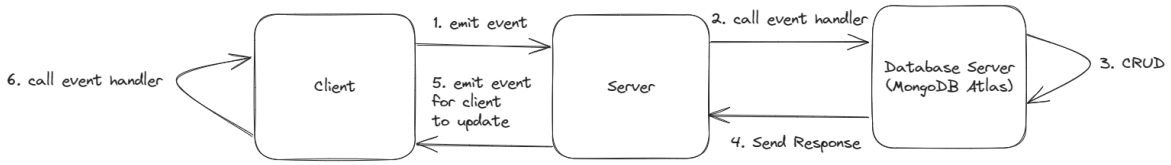
```
  $('#btn-logout').click(() => { logout(token) })

  $('#file').change(() => { previewUploadFile() })

  $('#inputText').on('change keydown paste input', () =>
{ typeMessage( conversationId, token)});

  socket.on('response-typing', (data) => {
responseTyping(data) })
```

3.2.4 Xây dựng cơ sở dữ liệu

- Accounts:
  - name (String): Tên tài khoản
  - email (String): Email tài khoản
  - password (String): Mật khẩu được mã hoá của tài khoản
  - avatar (String): Đường dẫn đến ảnh đại diện của tài khoản
  - isOnline (Boolean): Trạng thái của tài khoản (false: offline, true: online)
- Messages:
  - content (String): nội dung của tin nhắn hoặc chuỗi mã hóa base64 của ảnh hoặc file
  - accountID (ObjectId): id của người gửi
  - type (String: ‘file’, ‘image’, ‘text’): Loại tin nhắn
  - time (DateTime): Ngày gửi tin nhắn
  - fileName (String): Tên của file được gửi
- Conversations:
  - members (Array): mảng danh sách các id người dùng thuộc cuộc hội thoại

- messages (Array): mảnh danh sách các id tin nhắn thuộc cuộc hội thoại

3.2.5 Xây dựng chức năng



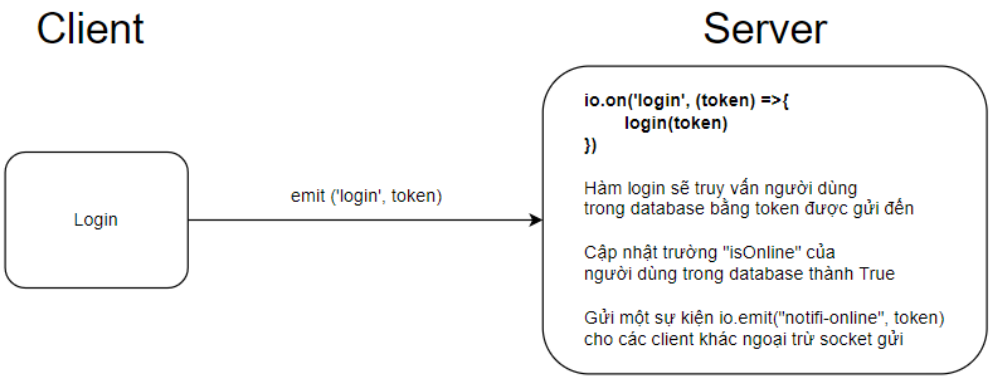
Hình 3.2.5: Luồng hoạt động các chức năng có sử dụng WebSocket

- Mô tả luồng hoạt động chung cho các chức năng có sử dụng WebSocket:
    - Bước 1: Client phát ra sự kiện
    - Bước 2: Server lắng nghe được sự kiện này và gọi hàm xử lý sự kiện tương ứng
    - Bước 3: Nếu hàm xử lý cần tương tác với cơ sở dữ liệu thì tiến hành lấy, thêm, sửa, xóa dữ liệu
    - Bước 4: Database Server trả response về server (app) của chúng em
    - Bước 5: Sau đó Server sẽ phát ra sự kiện để cho client có thể update lại
    - Bước 6: Client lắng nghe sự kiện được phát ra từ server và gọi hàm xử lý tương ứng

3.2.5.1 Đăng ký

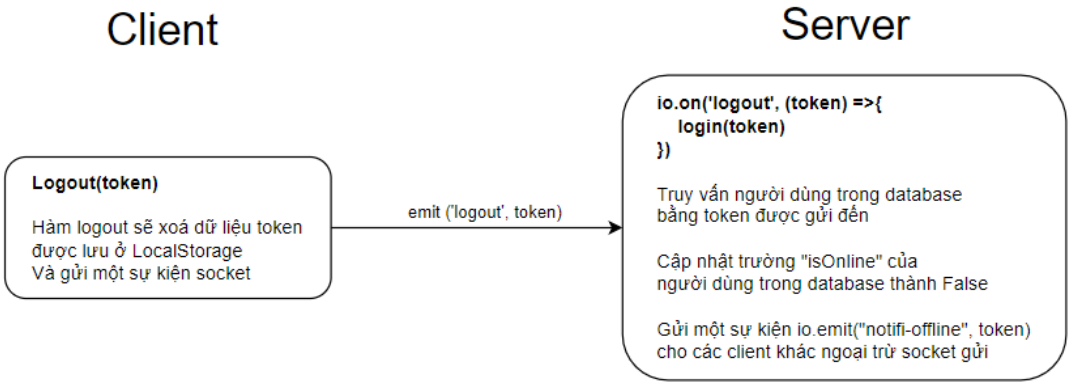
- Sau khi người dùng điền các thông tin cần thiết để tạo tài khoản và nhấn nút xác nhận.
- Hệ thống sẽ gửi một request đến server để lưu dữ liệu vào database.

3.2.5.2 Đăng nhập



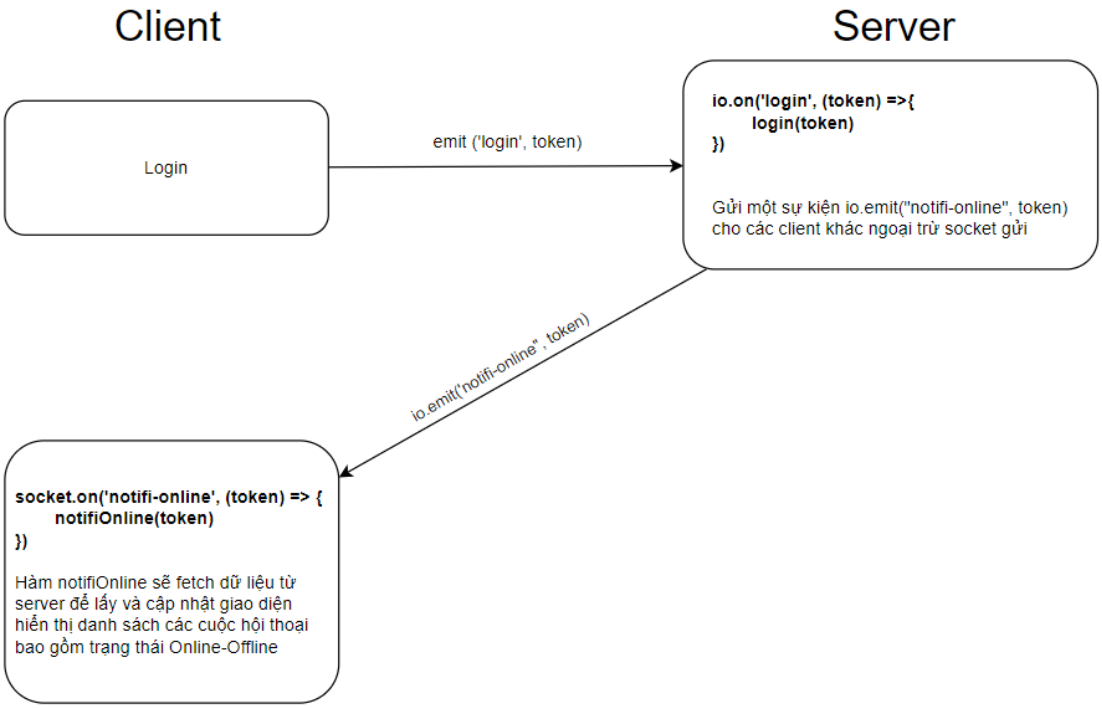
Hình 3.2.5.2: Đăng nhập

3.2.5.3 Đăng xuất



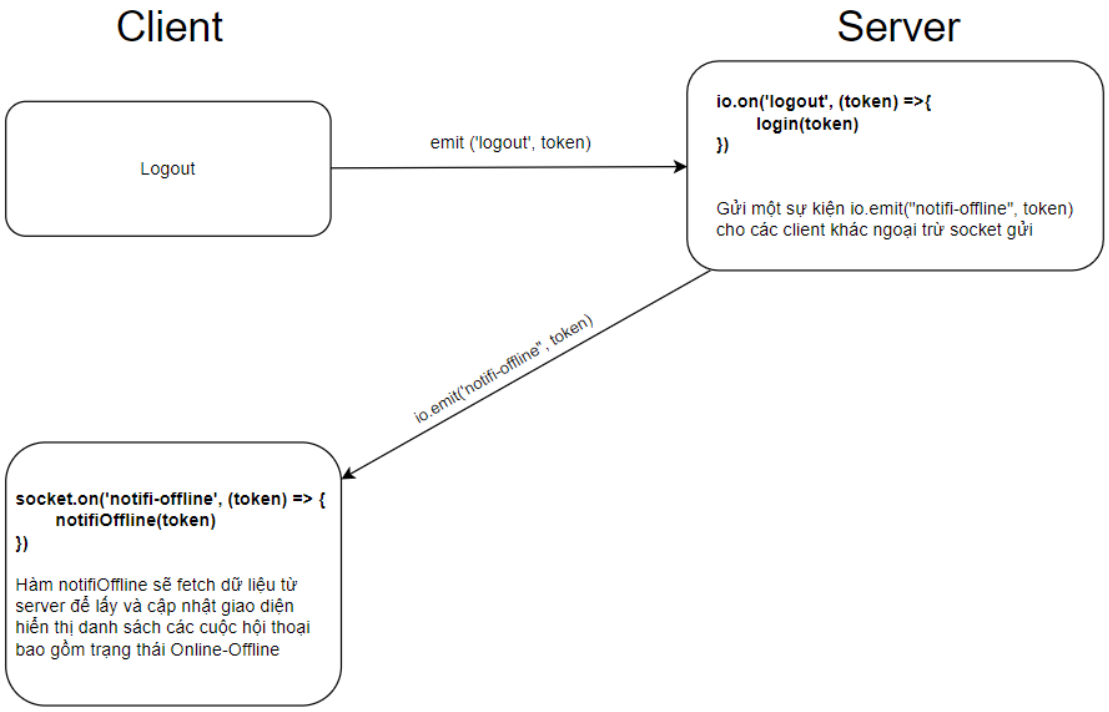
Hình 3.2.5.3: Đăng xuất

3.2.5.4 Hiển thị trạng thái online



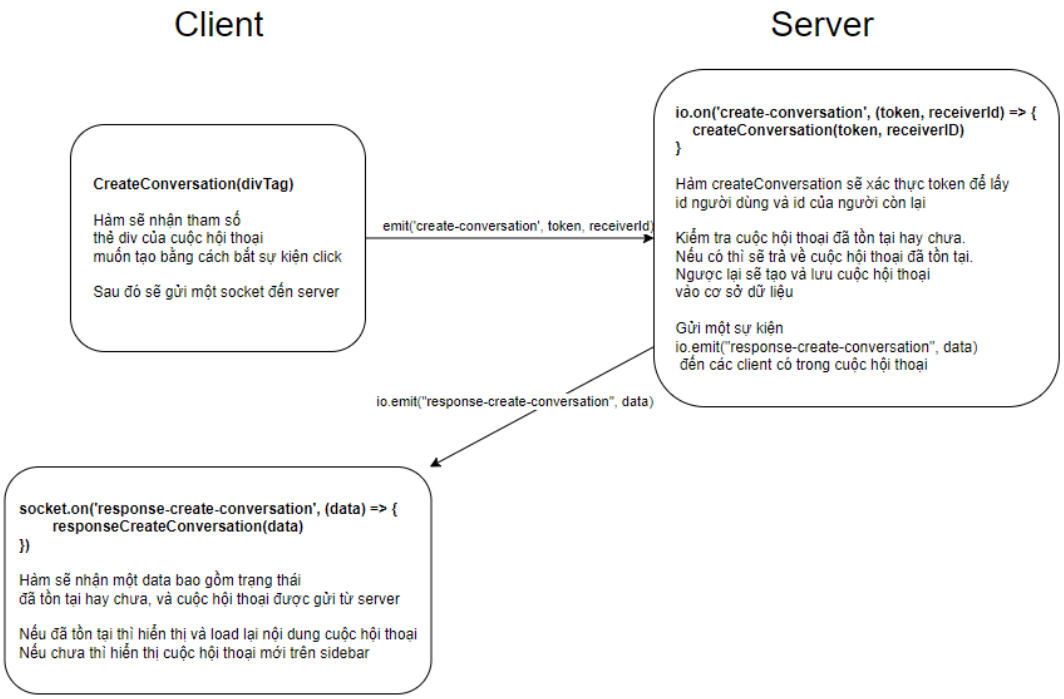
Hình 3.2.5.4: Hiển thị trạng thái online

3.2.5.5 Hiển thị trạng thái offline



Hình 3.2.5.5: Hiển thị trạng thái offline

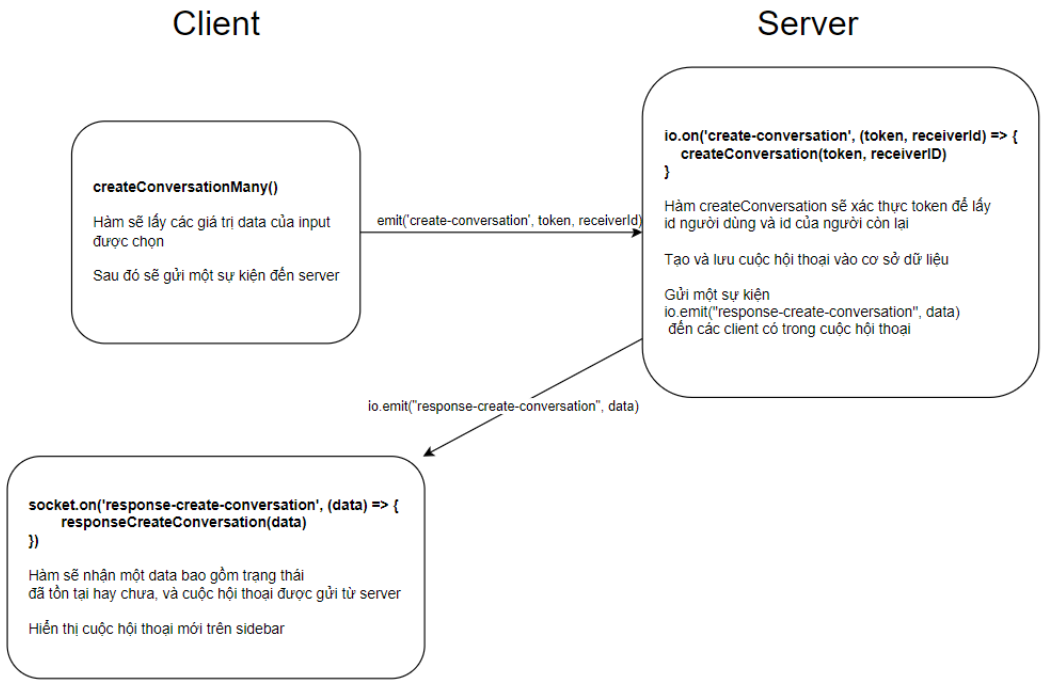
3.2.5.6 Tạo cuộc hội thoại cá nhân



Hình 3.2.5.6: Tạo cuộc hội thoại cá nhân

- Đối với cuộc hội thoại cá nhân, khi người gửi và người nhận đã có cuộc hội thoại trước đó thì khi nhấn tạo cuộc hội thoại mới giữa 2 người thì server sẽ tải lại nội dung cuộc hội thoại còn chưa thì sẽ tạo mới.

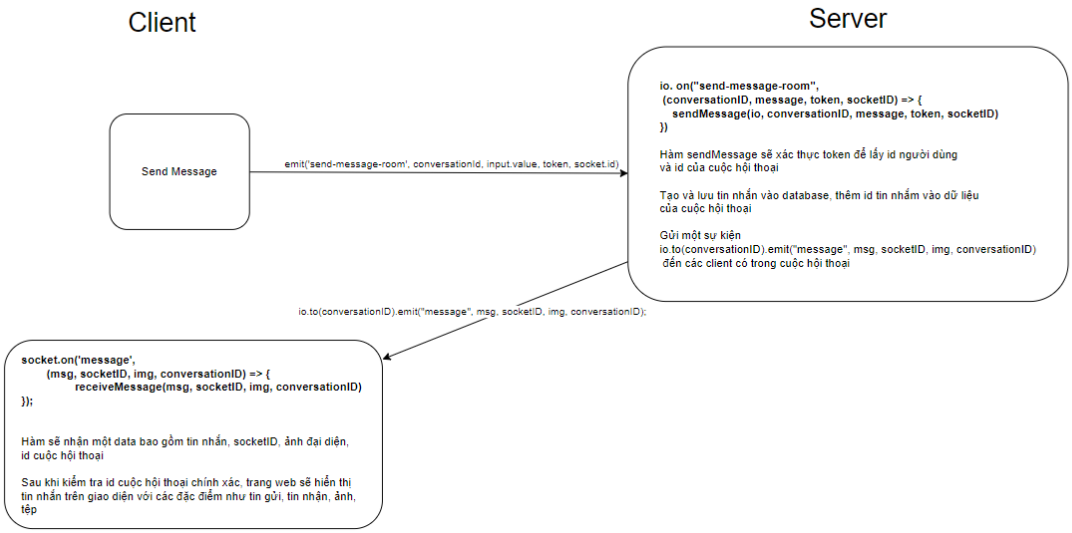
3.2.5.7 Tạo cuộc hội thoại nhóm



Hình 3.2.5.7: Tạo cuộc hội thoại nhóm

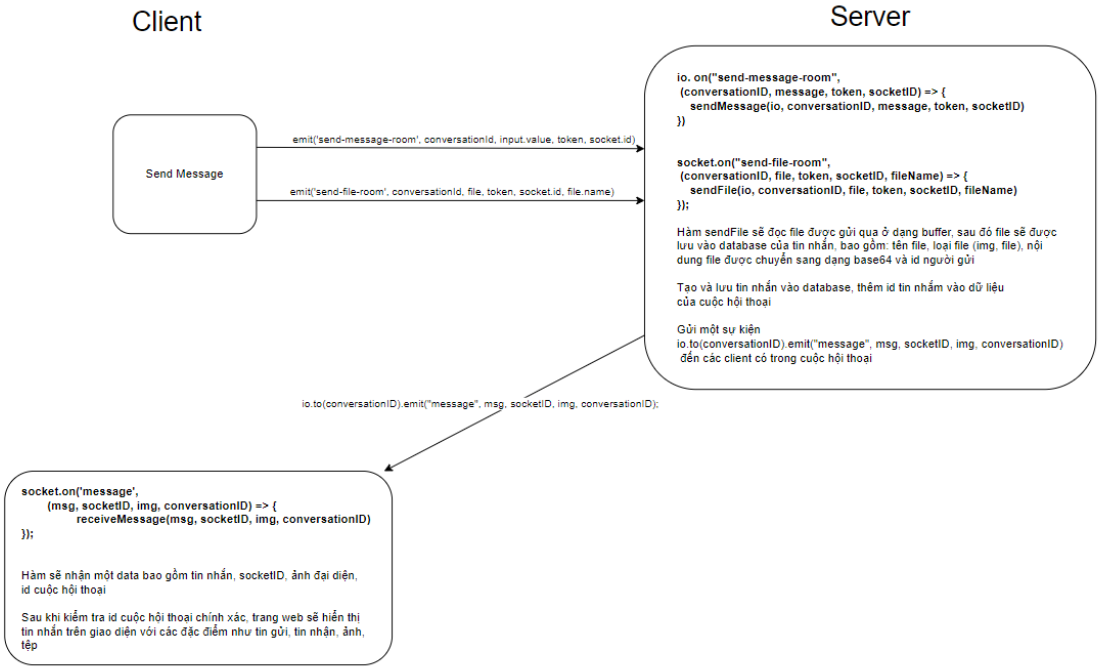
- Đối với các cuộc hội thoại nhóm, người dùng có thể tùy ý tạo nhiều nhóm.
- Ví dụ:
    - A, B, C là 3 người dùng của hệ thống. A, B, C trước đó đã có cuộc hội thoại nhóm gồm 3 người thì A, B, C có thể tiếp tục tạo ra các cuộc hội thoại nhóm khác bao gồm A, B, C.

3.2.5.8 Gửi và nhận tin nhắn



Hình 3.2.5.8: Gửi và nhận tin nhắn

3.2.5.9 Gửi hình ảnh, tệp



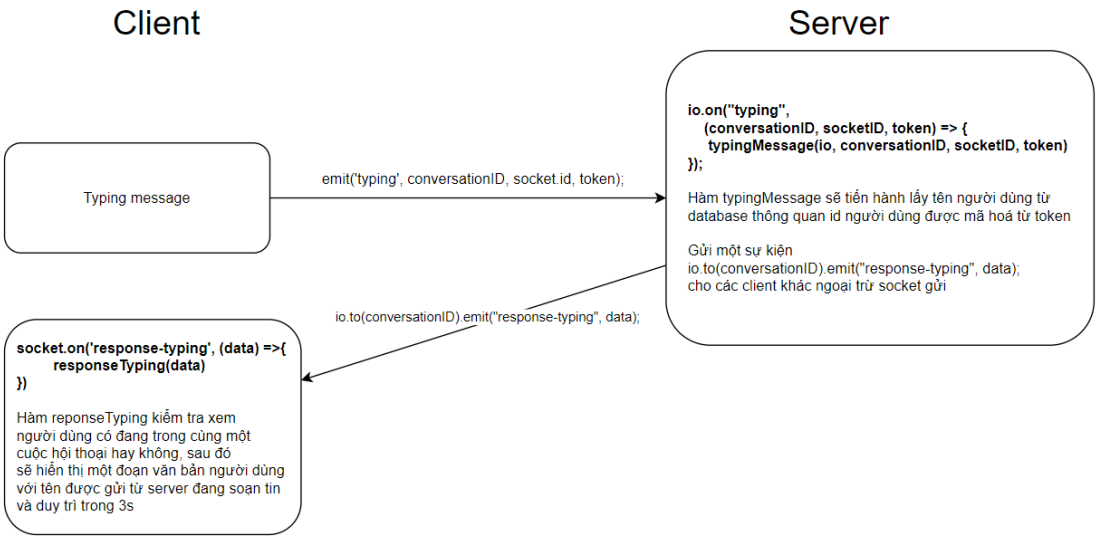
Hình 3.2.5.9: Gửi hình ảnh, tệp

3.2.5.10 Tải tệp

- Khi người dùng nhấn vào tin nhắn dạng tệp trên giao diện, hàm downloadFile() sẽ được kích hoạt bởi sự kiện onclick.
- Hàm downloadFile sẽ lấy data-id của thẻ div được click, sau đó tiến hành fetch dữ liệu của tệp thông qua id tin nhắn.
- Sau đó sẽ tiến hành tải tệp về trên máy người dùng thông qua:

```
const link = document.createElement('a');
link.href = `data:application/octet-stream;base64,${content}`;
link.download = fileName;
link.click();
```

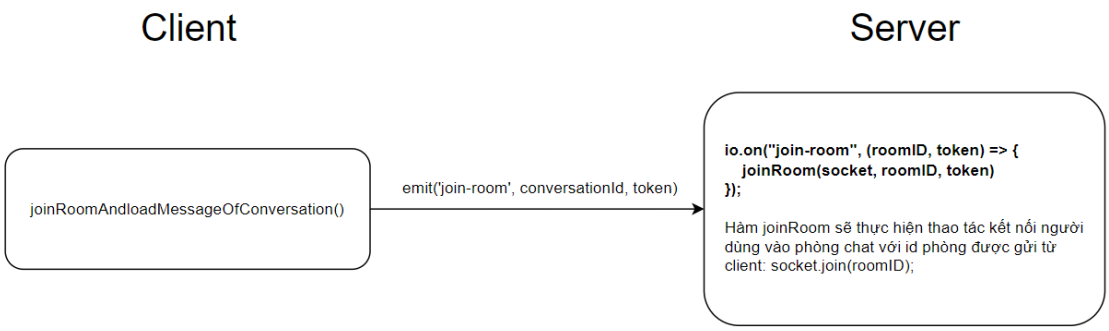
3.2.5.11 Hiển thị người dùng đang soạn tin nhắn



Hình 3.2.5.11: Hiển thị người dùng đang soạn tin nhắn

3.2.5.12 Tham gia vào phòng và hiển thị cuộc hội thoại cũ

- Khi người dùng nhấn vào tên cuộc hội thoại muốn liên lạc, hàm joinRoomAndloadMessageOfConversation() sẽ được kích hoạt bởi sự kiện.
- Hàm tiến hành fetch dữ liệu tin nhắn từ server, sau đó sẽ tạo giao diện cho các tin nhắn theo dạng tin nhắn văn bản, hình ảnh hoặc tệp.
- Đồng thời, từ client sẽ gửi một sự kiện socket đến server để người dùng kết nối phòng chat:



Hình 3.2.5.12: Tham gia vào phòng và hiển thị cuộc hội thoại cũ

TÀI LIỆU THAM KHẢO

Tiếng Việt:

TopDev. "Socket là gì? WebSocket là gì? Hiểu hơn về Websocket." [Online]. Available:https://topdev.vn/blog/socket-la-gi-websocket-la-gi-hieu-hon-ve-websocket/. [Accessed: 2023-12-03].

Tiếng Anh:

Ably Blog. "What are WebSockets used for? | Key use cases and who uses them today." [Online]. Available: https://ably.com/topic/what-are-websockets-used-for. [Accessed: 2023-12-03].

GeeksforGeeks. "What is Web Socket and how it is different from the HTTP?" [Online]. Available: https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/. [Accessed: 2023-12-03].

Stringee. "Websocket là gì? Ưu và nhược điểm của websocket bạn cần biết." [Online]. Available: https://stringee.com/vi/blog/post/websocket-la-gi. [Accessed: 2023-12-03].

Socket.IO. "Client API." [Online]. Available: https://socket.io/docs/v4/client-api/. [Accessed: 2023-12-03].