

Thực hành Buổi 7

LẬP TRÌNH SOCKET

I. LÝ THUYẾT CẦN ÔN TẬP VÀ THAM KHẢO

- Kết nối Socket
- NodeJs và Socket.io
- Lập trình Client Socket trên Android

II. THỰC HÀNH

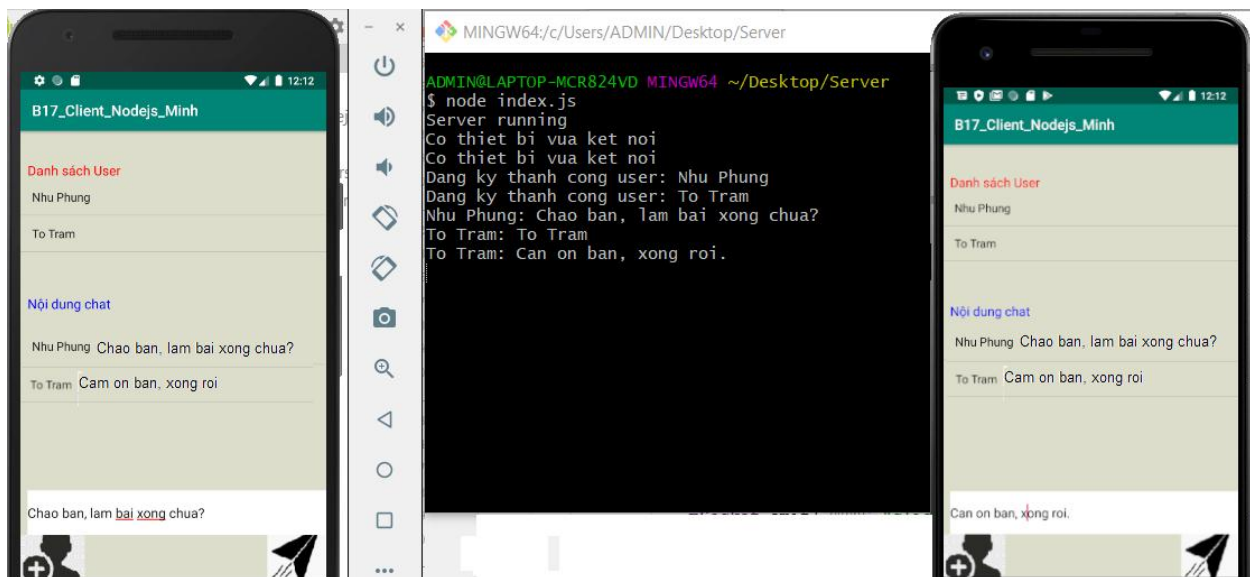
Ngày nay, những ứng dụng realtime như mạng xã hội, chat, game online, điều khiển từ xa... đang ngày càng phát triển thì công nghệ realtime luôn được ưa chuộng hơn bao giờ hết. Một trong những công nghệ phục vụ realtime mạnh mẽ nhất trong đó phải kể đến NodeJs và [Socket.io](https://socket.io/). Trong bài này, chúng ta sẽ thực hành về NodeJs và [Socket.io](https://socket.io/), viết một ứng dụng chat realtime đơn giản trên Android.

MÔ TẢ ỨNG DỤNG:

Chúng ta sẽ viết một ứng dụng chat giữa 2 client (điện thoại Android) thông qua Server (Nodejs), thực hiện các sự kiện sau:

1. Client gửi yêu cầu kết nối đến server: *connect*
2. Client đăng ký user
3. Server trả về kết quả đăng ký
4. Chat giữa 2 client thông qua server

Giao diện của Server (console của Git) và của Client (máy ảo Android) như Hình 1:



Hình 1

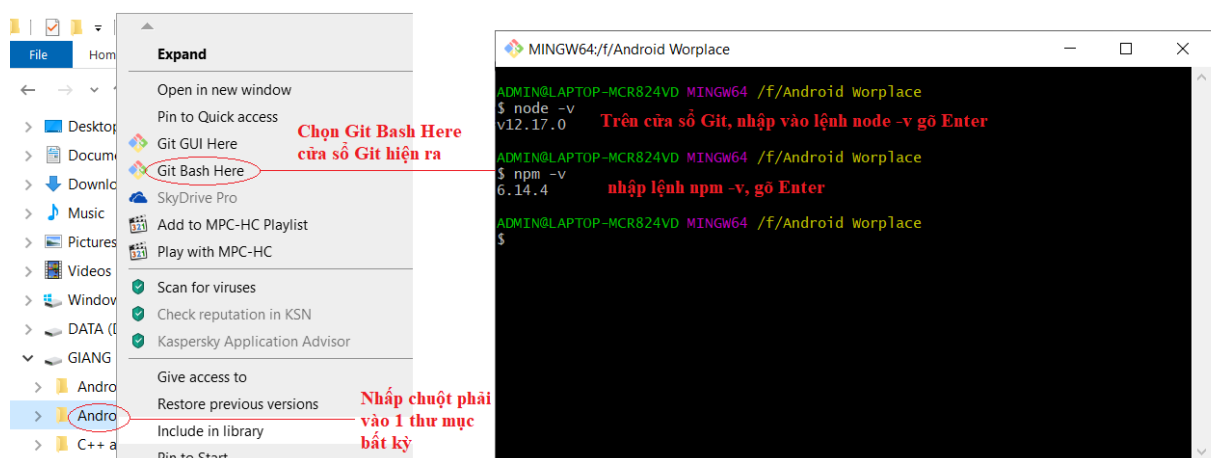
Mô hình hoạt động như Hình 2



Hình 2

CÀI ĐẶT CÁC PHẦN MỀM CẦN THIẾT

- 1) Cài đặt Nodejs, nếu chưa cài đặt, tải về và cài đặt theo mặc định tại link <https://nodejs.org/en/>.
- 2) Cài đặt Git, công cụ hỗ trợ chạy Server Nodejs (thay vì chạy với Command Promp), tải về và cài đặt theo mặc định tại link <https://git-scm.com/download/win>.
- 3) Kiểm tra kết quả cài đặt, nhấp chuột phải vào thư mục bất kỳ, chọn Git Bash Here (Hình 3). Để kiểm tra kết quả cài đặt Nodejs ta nhập vào lệnh **node -v** (có khoảng cách giữa node và -v), gõ enter, nếu cài đặt thành công thì số phiên bản nodejs hiện ra (ở đây là V12.17.0). Trong Nodejs có thành phần **npm** (NPM viết tắt của Node package manager là một công cụ tạo ra và quản lý các thư viện lập trình Javascript cho Node.js.), để kiểm tra, ta nhập lệnh **npm -v**, gõ enter, nếu cài đặt thành công thì số phiên bản cũng hiện ra.



Hình 3

CÁC BƯỚC THỰC HIỆN

- 1) **Tạo thư mục chứa server:** Tạo thư mục mới (trong hướng dẫn này đã chọn trên Desktop), đặt tên là **Server**.
- 2) **Tạo tập tin package.json trên thư mục Sever:** Có nhiều cách thực hiện, trong hướng dẫn này ta làm như sau: mở Notepad, chưa cần soạn thảo nội dung (hoặc soạn thảo luôn trên Notepad vẫn được) ta Save As vào thư mục Server với tên và phần đuôi là **package.json**. Sau đó thoát khỏi Notepad và mở thư mục Server (nhấp đúp chuột vào thư mục Server) → nhấp chuột phải vào tập tin package.json → chọn **Open with Sublime Text** để soạn thảo nội dung của tập tin này, code như sau:

```
{
  "name": "Server",
  "version": "0.0.1",
  "private": "true",
  "dependencies": {
    "express": "*",
    "mysql": "^2.14.1",
    "socket.io": "*",
    "socketio-file-upload": "^0.4.4"
  }
}
```

Trong đó:






- name: là tên thư mục.
- Trong cặp Dependencies là những thư viện cần thiết của server, trong phạm vi bài này ta có thể không cần đến mysql (có thể xóa lệnh này).

- 3) **Tạo tập tin index.html trên thư mục Sever:** Đây là tập tin HTML dùng để truy cập thử vào server. Ta thực hiện như bước 2 (nhưng lưu tập tin với đuôi là html), nội dung tập tin đơn giản như sau:

```
<h1> Amdroid NodeJs </h1>
```

- 4) **Chạy tập tin package.json để tải về thư viện Socket.io:** Nhấp chuột phải vào tên tập tin → Chọn **Git Bash Here** → Màn hình Git mở ra, nhập lệnh **npm install** vào và gõ enter → Chờ vài phút để các module của nodejs được tải về.

Sau khi tải các module thành công sẽ xuất hiện thư mục **node_modules** (khoảng 5.5MB). Thư mục Server đến lúc này gồm các tập tin và thư mục như sau:

Name	Date modified	Size	Type
 package-lock	6/2/2020 6:32 PM	31 KB	JSON File
 index	6/4/2020 10:45 AM	1 KB	JavaScript File
 package	6/2/2020 6:32 PM	1 KB	JSON File
 index	6/2/2020 9:01 PM	1 KB	CocCoc HTML Doc...
 node_modules	6/2/2020 6:32 PM		File folder

- 5) **Cấu hình Express:** Tạo tập tin mới và viết mã nguồn cho server (bằng ngôn ngữ JavaScript), trong hướng dẫn này ta đặt tên tập tin là **index.js**. Ta cũng thực hiện như bước 2, mở Notepad, chưa cần soạn thảo nội dung, ta Save As vào thư mục Server với tên và phần đuôi là **index.js**. Sau đó thoát khỏi Notepad và mở thư mục Server (nhấp đúp chuột vào thư mục Server) → nhấp chuột phải vào tập tin **index.js** → chọn **Open with Sublime Text** để soạn thảo nội dung của tập tin này, đây là code của server.

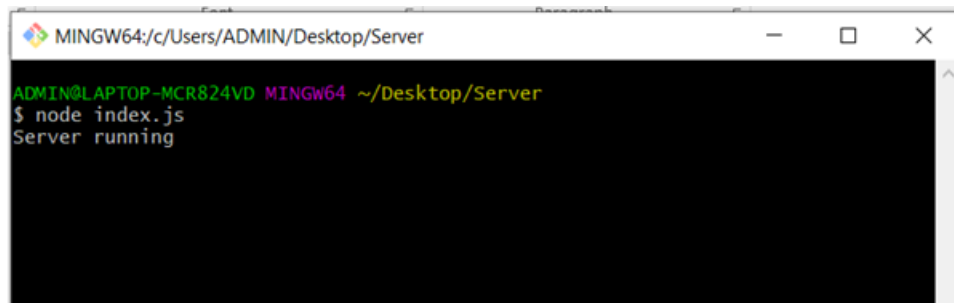
Trước tiên, ta viết tạm một phần để kiểm thử, trong các bước sau sẽ hoàn chỉnh dần. Nội dung như sau:

```
var express = require("express");
var app = express();
var server = require("http").createServer(app);
var io = require("socket.io").listen(server);
var fs = require("fs");
server.listen(process.env.PORT || 3000);
app.get("/", function(req, res){
    res.sendFile(__dirname + "/index.html");
});
console.log("Server running");
```

Trong đó:

- 5 dòng đầu là khai báo các biến, gọi các thư viện.
- Dòng thứ 6, ấn định sever lắng nghe ở port 3000 (có thể đổi port khác).
- Dòng 7, 8 và 9 chỉ định mở 1 tập tin HTML mặc định khi truy cập vào server (localhost) mà chúng ta đã tạo ra trong thư mục Server, dùng để kiểm thử Server trước khi thực hiện các bước kế tiếp.
- Dòng 10, báo trên console của Git là Server đang chạy.

- 6) **Chạy thử Server:** Mở lại màn hình Git (đóng màn hình Git cũ) bằng cách **nhấp chuột phải và thư mục Server**, chọn Git Bash Here, nhập lệnh **node index.js** và gõ enter. Server chạy, trên console của Git hiển thị như Hình 4. Để thử, ta mở trình duyệt vào localhost với địa chỉ <https://localhost:3000>, trang web của Server được mở vào hiển thị nội dung của tập tin HTML index.html như Hình 5.



Hình 4



Hình 5

7) Trên Android Studio ta mở Project mới để lập trình Client Socket:

Trong hướng dẫn này ta đặt tên project là B17_Socket_MSSV_Tên SV với activity chính là MainActivity.java và Layout chính là activity_main.xml.

8) Cấu hình các modules cho phía client:

- Để kết nối phía Client đến Server, ta cần các thư viện hỗ trợ. Để tìm các thư viện này, ta search với từ khóa Android Socket IO → Chọn GitHub – Socketio/socket.io-c, với actlient-java: Full=Featured Socket... chọn mục project → đến trang web có URL là <https://github.com/socketio/socket.io-client-java> (hoặc đăng nhập trực tiếp với đường dẫn này). Trong trang web này, ta đến mục Gradle như Hình 6

Gradle

Add it as a gradle dependency for Android Studio, in `build.gradle` :

```
compile ('io.socket:socket.io-client:1.0.0') {
    // excluding org.json which is provided by Android
    exclude group: 'org.json', module: 'json'
}
```

Hình 6

Ta copy nội dung sau đây:

```
compile ('io.socket:socket.io-client:1.0.0') {
    // excluding org.json which is provided by Android
    exclude group: 'org.json', module: 'json'
}
```

và paste vào mục **dependencies** của tập tin **build.gradle** (Module:app) trong thư mục **Gradle Scripts** của project, đồng thời thêm vào dòng lệnh `implementation 'com.android.support:recyclerview-v7:27.0.2'` (để khi cần tạo giao diện có thành phần recyclerview), như sau:

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    compile ('io.socket:socket.io-client:1.0.0') {
        // excluding org.json which is provided by Android
        exclude group: 'org.json', module: 'json'
    }
    implementation 'com.android.support:recyclerview-v7:27.0.2'
}
```

- Thêm quyền truy cập Internet cho ứng dụng của bạn trong file AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET"/>
```

9) Sửa code trên Server (Tập tin index.js, ngôn ngữ Java Script)

```
var express = require("express");
var app = express();
var server = require("http").createServer(app);
var io = require("socket.io").listen(server);
var fs = require("fs");
server.listen(process.env.PORT || 3000);

// Cho lệnh truy cập tập tin index.html thành chú giải
//app.get("/", function(req, res){
//    res.sendFile(__dirname + "/index.html");
//});

// Hiển thị trên màn hình console cho biết server đang chạy
console.log("Server running");

// Server lắng nghe và chấp nhận yêu cầu kết nối từ client.
io.sockets.on('connection', function(socket){
    console.log("Có thiết bị vừa kết nối");
    socket.on('client-send-data', function(data){
        console.log("Server nhận: " + data);
        io.sockets.emit('server-send-data', {noidung : data});
    });
});
```

Trên server ta đã dùng mảng các để lưu các user đã login

- Sự kiện `io.sockets.on('connection', function (socket)` lắng nghe các sự kiện kết nối lên server
- Sự kiện `socket.on('client-send-data', function(data)` lắng nghe các sự kiện client gửi lên server với data là data.
- Sự kiện `io.sockets.emit('server-send-data', {noidung: data});` gửi data đến client

10) Viết code cho client: Trong bước này ta chưa tạo layout, viết tạm code cho MainActivity để chạy thử (*thử từng bước để dễ phát hiện lỗi và sửa lỗi*). Code như sau:

```
package com.example.b17_client_nodejs_minh;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.Toast;
import org.json.JSONException;
import org.json.JSONObject;
import java.net.URISyntaxException;
import io.socket.client.IO;
import io.socket.client.Socket;
import io.socket.emitter.Emitter;

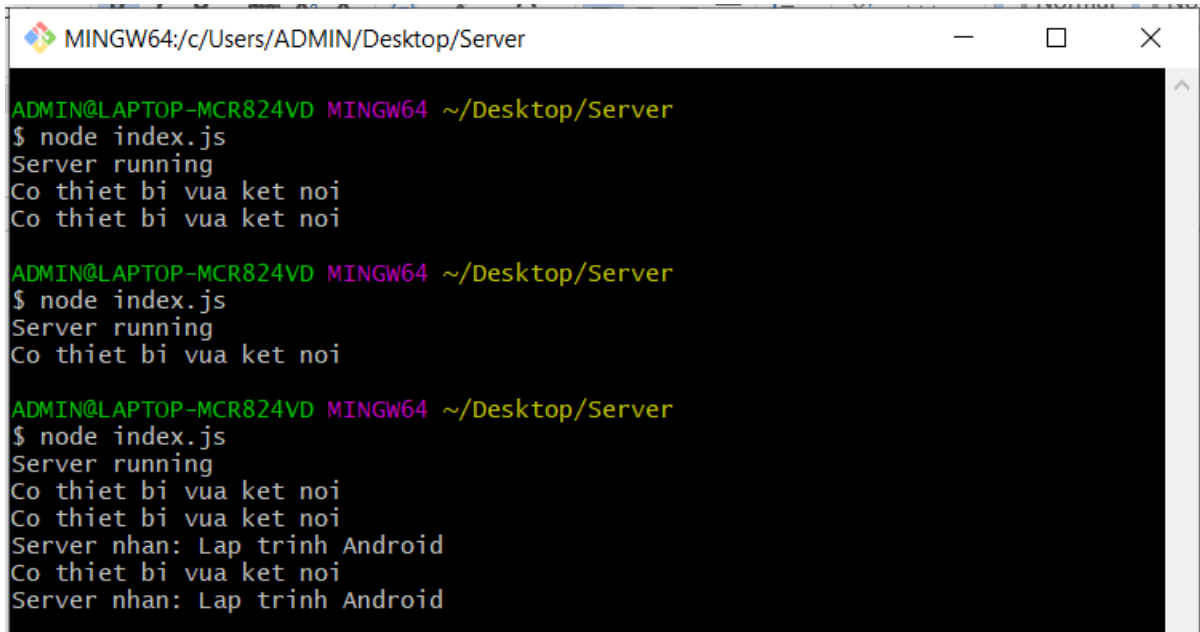
public class MainActivity extends AppCompatActivity {
    private final String URL_SERVER = "http://192.168.1.37:3000";
    private TextView mTVRequest;
    private Socket mSocket; // Chọn Socket (IO.socket.client)

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        try {
            mSocket = IO.socket(URL_SERVER);
        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
        mSocket.connect();
        mSocket.on("server-send-data", onRetrieveData);
        mSocket.emit("client-send-data", "Lập trình Android");
    }

    private Emitter.Listener onRetrieveData = new Emitter.Listener() {
        @Override
        public void call(final Object... args) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    JSONObject object = (JSONObject) args[0];
                    try {
                        String ten = object.getString("noidung");
                        Toast.makeText(MainActivity.this, ten, Toast.LENGTH_LONG).show();
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            });
        }
    };
}
```

```
};  
}
```

- 11) **Chạy thử:** Trở lại console của Git ta nhấn **ctrl+c** để tắt Server và khởi chạy lại Server với lệnh **node index.js**. Trên Android Studio ta chạy Client với máy ảo. Kết quả trên máy ảo hiện Toast với text là “Lap trình Android” và Sever hiển thị trên console của Git như Hình 7.

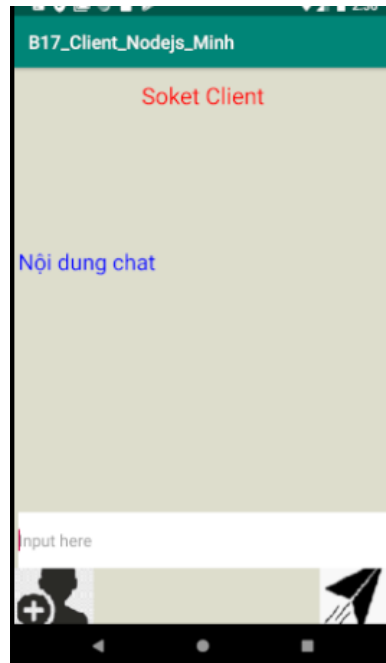


```
MINGW64:/c/Users/ADMIN/Desktop/Server  
ADMIN@LAPTOP-MCR824VD MINGW64 ~/Desktop/Server  
$ node index.js  
Server running  
Co thiet bi vua ket noi  
Co thiet bi vua ket noi  
  
ADMIN@LAPTOP-MCR824VD MINGW64 ~/Desktop/Server  
$ node index.js  
Server running  
Co thiet bi vua ket noi  
  
ADMIN@LAPTOP-MCR824VD MINGW64 ~/Desktop/Server  
$ node index.js  
Server running  
Co thiet bi vua ket noi  
Co thiet bi vua ket noi  
Server nhan: Lap trinh Android  
Co thiet bi vua ket noi  
Server nhan: Lap trinh Android
```

Hình 7

Ghi chú: Nếu màn hình console Git đầy nội dung, ta dùng lệnh **clear** để xóa.

- 12) **Viết lại giao diện cho Client:** Đến bước 11, ta đã thực hiện thành công một Server Nodejs hoạt động localhost và ứng dụng Client trên Android, thực hiện các hoạt động: Server lắng nghe, Client gửi yêu cầu kết nối và Server chấp nhận kết nối, Client gửi thông điệp, Server nhận và trả về Client. Đến bước này, chúng ta sẽ cải tiến hệ thống thành ứng dụng chat đơn giản. Trước tiên ta viết code layout cho Client. Giao diện gồm các thành phần như Hình 8.



Hình 8

- Tập tin giao diện xml

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="10"
    android:background="#ddc"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Soket Client"
        android:gravity="center"
        android:layout_gravity="center"
        android:textColor="#f00"
        android:textSize="25sp" />

    <ListView
        android:id="@+id/listviewUser"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2">

    </ListView>

    <TextView
        android:layout_weight="1"
        android:gravity="center_vertical"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:text="Nội dung chat"
        android:textColor="#00f"
        android:textSize="24sp"
        android:layout_marginLeft="5dp"/>

    <ListView
        android:id="@+id/listviewChat"
        android:layout_width="match_parent"
```

```

        android:layout_height="0dp"
        android:layout_weight="5">
    </ListView>
    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">
        <TableRow>
            <EditText
                android:layout_width="0dp"
                android:layout_weight="6"
                android:layout_height="match_parent"
                android:id="@+id/editTextContent"
                android:hint="Input here"
                android:background="#fff"
                android:layout_marginLeft="5dp"/>
            <ImageButton
                android:id="@+id/btn_login"
                android:layout_width="0dp"
                android:layout_weight="2"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:background="@null"
                android:src="@drawable/dangky_1"/>

            <ImageButton
                android:id="@+id/btnchat"
                android:layout_width="0dp"
                android:layout_weight="2"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:background="@null"
                android:src="@drawable/icon_send_1"/>
        </TableRow>
    </TableLayout>
</LinearLayout>

```

13) Sửa lại code của MainActivity.java đồng thời với code của Server

- Mã nguồn tham khảo của MainActivity

```

age com.example.b17_client_nodejs_minh;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONStringer;

import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.List;

import io.socket.client.IO;

```

```

import io.socket.client.Socket;
import io.socket.emitter.Emitter;

public class MainActivity extends AppCompatActivity {
    // Khai báo các View
    private Button mButtonLogin;
    private Button mButtonChat;
    private EditText edtContent;
    private ListView lvUser, lvChat;
    private ImageButton btnAdd, btnSend;
    private TextView mTVRequest;
    ArrayList<String> arrayUser, arrayChat;
    ArrayAdapter adapterUser, adapterChat;
    // Khai báo ip và port của Server
    // Gõ lệnh ipconfig trên Command Prompt hoặc mở Tark Manager để tìm ip.
    private final String URL_SERVER = "http://192.168.1.37:3000";
    private Socket mSocket; // Chọn Socket (IO.socket.client)

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Ánh xạ các view từ layout
        btnAdd = findViewById(R.id.btn_login);
        btnSend = findViewById(R.id.btnchat);
        edtContent = findViewById(R.id.editTextContent);
        lvChat = findViewById(R.id.listviewChat);
        lvUser = findViewById(R.id.listviewUser);
        // Khởi tạo arrayUser, adapterUser và đưa arrayUser vào ListView
        arrayUser = new ArrayList<>();
        adapterUser = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
arrayUser);
        lvUser.setAdapter(adapterUser);
        // Khởi tạo arrayChat, adapterChat và đưa arrayChat vào ListView
        arrayChat = new ArrayList<>();
        adapterChat = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
arrayChat);
        lvChat.setAdapter(adapterChat);
        // Gọi yêu cầu kết nối
        try {
            mSocket = IO.socket(URL_SERVER);
        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
        mSocket.connect();
        mSocket.on("server-send-data", onRetrieveResult);
        mSocket.on("server-send-user", onListUser);
        mSocket.on("server-send-chat", onListChat);

        // Xử lý tương tác cho nút đăng ký user.
        btnAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Nếu có nhập tên vào EditText thì gọi đăng ký User.
                if(edtContent.getText().toString().trim().length()>0){
                    mSocket.emit("client-register-user",
edtContent.getText().toString());
                    // Nhớ sửa lại sự kiện bên Server
                }
            }
        });

        btnSend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Nếu có nhập message vào EditText thì gọi.
                if (edtContent.getText().toString().trim().length() > 0) {

```

```

        mSocket.emit("client-send-chat",
edtContent.getText().toString());
        // Nhớ sửa lại sự kiện bên Server
    }
}
});
}
private Emitter.Listener onListChat = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                JSONObject object = (JSONObject) args[0];
                try {
                    String noiDung = object.getString("chatComent");
                    arrayChat.add(noiDung);
                    adapterUser.notifyDataSetChanged();
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        });
    }
};

private Emitter.Listener onListUser = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                JSONObject object = (JSONObject) args[0];
                try {
                    JSONArray array = object.getJSONArray("danhsach");
                    adapterUser.clear();
                    for (int i = 0; i < array.length(); i++) {
                        String username = array.getString(i);
                        adapterUser.add(username);
                    }
                    adapterUser.notifyDataSetChanged();
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        });
    }
};

private Emitter.Listener onRetrieveResult = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                JSONObject object = (JSONObject) args[0];
                try {
                    //String ten = object.getString("noidung");
                    boolean exits = object.getBoolean("ketqua");
                    if(exits) {
                        Toast.makeText(MainActivity.this, "Tài khoản này đã tồn
tại!", Toast.LENGTH_LONG).show();
                    } else {
                        Toast.makeText(MainActivity.this, "Đã đăng ký thành
công", Toast.LENGTH_LONG).show();
                    }
                } catch (JSONException e) {

```

```

        e.printStackTrace();
    }
}
});
}
};
}

```

- **Mã nguồn Sever (tập tin index.js) tham khảo**

```

var express = require("express");
var app = express();
var server = require("http").createServer(app);
var io = require("socket.io").listen(server);
var fs = require("fs");
server.listen(process.env.PORT || 3000);
//app.get("/", function(req, res){
    //res.sendFile(__dirname + "/index.html");
//});

console.log("Server running");
var arrayUser = []; // Mảng chứa các User
var tontai = true; // Biến để kiểm tra user có tồn tại không
io.sockets.on('connection', function(socket){
    console.log("Co thiet bi vua ket noi");

    socket.on('client-register-user', function(data){
        if (arrayUser.indexOf(data)== -1) {
            // Không tồn tại User trong danh sách đã đăng ký
            arrayUser.push(data);
            tontai = false;
            console.log("Dang ky thanh cong user: " + data);
            // Gán tên user cho client socket, data chính là tên user.
            socket.un = data; //hàm un gán tên lại cho socket.
            // Gửi danh sách user về tất cả các máy.
            io.sockets.emit('server-send-user', { danhsach : arrayUser});
        } else {
            console.log("Da ton tai user: " + data);
            tontai = true;
        }
        // Gửi kết quả đăng ký đến user đã đăng ký (1 client).
        io.sockets.emit('server-send-data', {ketqua : tontai});
    });

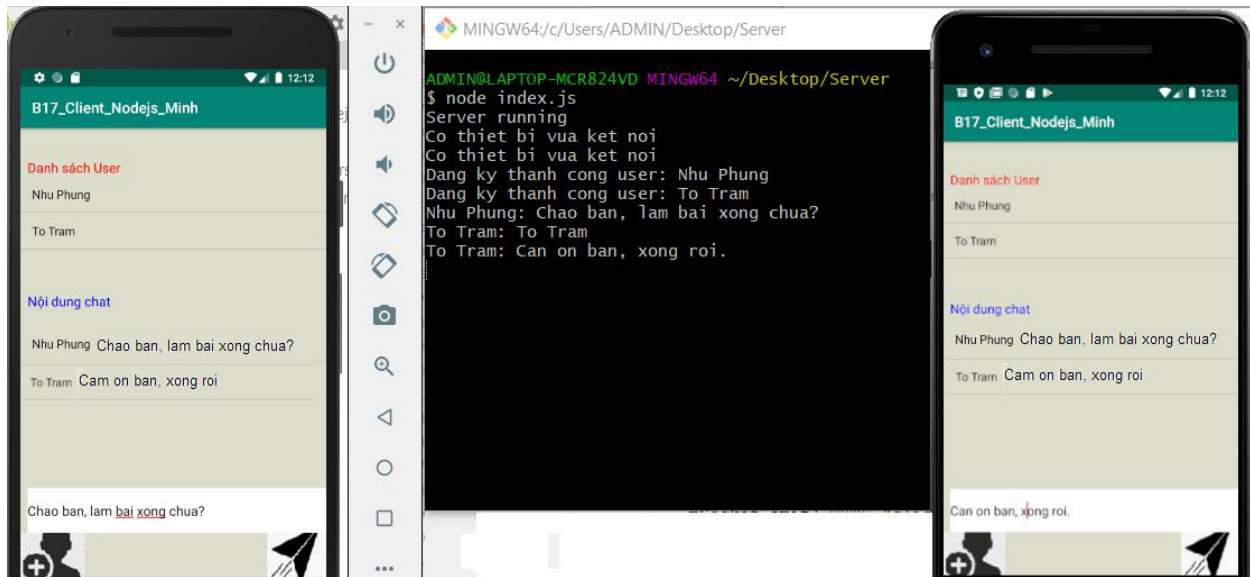
    socket.on('client-send-chat', function(noiDung){
        console.log(socket.un + ": " + noiDung);
        io.sockets.emit('server-send-chat', { chatComent : socket.un + ": " + noiDung});
    });

});

```

14) Chạy thử:

- Server: Trên màn hình Git, ta nhập lệnh **clear** nhấn enter để xóa màn hình, sau đó nhấn tổ hợp **Ctl+C** để tắt Server và nhập lệnh **node index.js** nhấn enter để khởi chạy lại Server.
- Trên Android Studio ta chạy project (client) với 2 máy ảo:
 - Nhập tên (vào EditText) và nhấn nút Đăng ký (góc trái dưới – hình người +) để đăng ký user cho từng máy.
 - Nhập tin nhắn (cũng vào cùng EditText đã nhập tên user) và nhấn nút Send (góc phải dưới – hình mũi tên) để gửi tin nhắn chat.
 - Kết quả hiển thị trên màn hình console của Git và các máy ảo như Hình 9.



Hình 9

Ghi chú:

- Trong mã nguồn tham khảo có chỗ sai, nên câu chat không hiện lên điện thoại ảo như hình 9, SV nghiên cứu sửa lại.
- Có thể chạy thử bằng 2 điện thoại thật, nhưng phải kết nối cùng wifi.
- Để tìm ip của localhost ta gõ lệnh ipconfig trên Command prompt.

Tài liệu tham khảo

- 1) Lập trình Android A-Z - Bài 173: <https://www.youtube.com/watch?v=1dDDerJAJd4>
- 2) Lập trình Android A-Z - Bài 174: <https://www.youtube.com/watch?v=nF31i3-nzPc>
- 3) Lập trình Android A-Z - Bài 175: <https://www.youtube.com/watch?v=dAmNhFn4S6g>
- 4) Lập trình Android A-Z - Bài 176: <https://www.youtube.com/watch?v=izYzTJABuII>
- 5) Lập trình Android A-Z - Bài 177: <https://www.youtube.com/watch?v=SbclccBoBwQ>
- 6) Lập trình Android A-Z - Bài 178: <https://www.youtube.com/watch?v=Pz6D1MS8iMU>
- 7) Lập trình Android A-Z - Bài 179: <https://www.youtube.com/watch?v=rhgtRoG4HyU>
- 8) <https://viblo.asia/p/xay-dung-ung-dung-chat-tren-android-voi-nodejs-va-socketio-phan-1-gDVK28WjLj>
- 9) <https://khoapham.vn/KhoaPhamTraining/android/snippet/>