REPUBLIQUE DU SENEGAL

UN PEUPLE-UN BUT-UNE FOI

EC2LT
ECOLE CENTRALE DES LOGICIELS LIBRES
ET DE TELECOMMUNICATIONS

RESEAUX ET TELECOMMUNICATIONS

OPTION : CYBERSECURITE/DEVOPS

Projet

Développement des services dans l'environnement télécoms

Rédiger par :

Serge Elvis Espoir BOUNGUELE

Sous la direction de :

Pr. Samuel OUYA

Année Académique 2024-2025

SOMMAIRE

INTRODUCTION

Ce projet vise à développer une application de visioconférence intégrant des fonctionnalités de conférence, chat et partage d'écran dans un environnement télécoms. En s'appuyant sur des technologies modernes telles que **WebRTC**, **PeerJS**, **Socket.IO** et Electron, l'objectif est de concevoir une solution robuste et performante, capable de fonctionner sur des plateformes web et desktop (**Windows** et **Linux**). Ce document détaille les étapes de développement, de la mise en place d'une application WebRTC de base à la création d'une application desktop complète avec gestion de comptes et base de données, tout en assurant une méthodologie claire pour la génération d'exécutables.

1. Application WebRTC (Apache2 et PeerJS)

1.1 Prérequis et Installation des Dépendances

1.1.1 Mise à jour du système

sudo apt update && sudo apt upgrade -y

```
root@sergio:/home/sergio# sudo apt update && sudo apt upgrade -y
sudo: impossible de résoudre l'hôte sergio: Échec temporaire dans la résolution du nom
Atteint :1 http://sn.archive.ubuntu.com/ubuntu focal InRelease
Atteint :2 https://deb.nodesource.com/node_20.x nodistro InRelease
Réception de :3 https://packages.grafana.com/enterprise/deb stable InRelease [7661 B]
Réception de :4 http://sn.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Réception de :5 https://download.docker.com/linux/ubuntu focal InRelease [57,7 kB]
```

1.1.2 Installation de Node.js et npm

curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -

```
root@sergio:/home/sergio# curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo: impossible de résoudre l'hôte sergio: Échec temporaire dans la résolution du nom
2025-06-23 01:31:00 - Installing pre-requisites
Atteint :1 http://sn.archive.ubuntu.com/ubuntu focal InRelease
Réception de :2 http://sn.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Atteint :3 https://deb.nodesource.com/node_20.x nodistro InRelease
Atteint :4 http://ppa.launchpad.net/oisf/suricata-stable/ubuntu focal InRelease
```

sudo apt-get install -y nodejs

```
root@sergio:/home/sergio# sudo apt-get install -y nodejs
sudo: impossible de résoudre l'hôte sergio: Échec temporaire dans la résolution du nom
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
nodejs est déjà la version la plus récente (20.19.2-1nodesource1).
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
```

1.1.3 Vérification

node -v

npm -v

```
root@sergio:/home/sergio# node -v
v20.19.2
root@sergio:/home/sergio# npm -v
10.8.2
root@sergio:/home/sergio#
```

1.1.4 Installation d'Apache2

sudo apt install apache2 -y

```
root@sergio:/home/sergio# sudo apt install apache2 -y
sudo: impossible de résoudre l'hôte sergio: Échec temporaire dans la résolution du nom
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
apache2 est déjà la version la plus récente (2.4.41-4ubuntu3.23).
```

systemctl start apache2

```
root@sergio:/home/sergio# systemctl start apache2
root@sergio:/home/sergio#
```

systemctl enable apache2

```
root@sergio:/home/sergio# systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable apache2
root@sergio:/home/sergio#
```

1.1.5 Vérification

systemctl status apache2

```
root@sergio:/home/sergio# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-06-22 23:18:01 GMT; 2h 24min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 988 (apache2)
      Tasks: 55 (limit: 9293)
     Memory: 7.5M
     CGroup: /system.slice/apache2.service
             ├─988 /usr/sbin/apache2 -k start
             ├─990 /usr/sbin/apache2 -k start
             └─991 /usr/sbin/apache2 -k start
```

1.2 Installation des outils de développement

sudo apt install git curl wget build-essential -y

```
root@sergio:/home/sergio# sudo apt install git curl wget build-essential -y
sudo: impossible de résoudre l'hôte sergio: Échec temporaire dans la résolution du nom
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
build-essential est déjà la version la plus récente (12.8ubuntu1.1).
build-essential passé en « installé manuellement ».
curl est déjà la version la plus récente (7.68.0-1ubuntu2.25).
git est déjà la version la plus récente (1:2.25.1-1ubuntu3.14).
wget est déjà la version la plus récente (1.20.3-1ubuntu2.1).
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  mongodb-database-tools mongodb-mongosh
Veuillez utiliser « sudo apt autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

1.3 Configuration d'Apache2 pour WebRTC

1.3.1 Création du répertoire du projet

On créer le dossier de l'application comme suite

sudo mkdir -p /var/www/html/webrtc-app

sudo chown -R $USER:$USER /var/www/html/webrtc-app

sudo chmod -R 755 /var/www/html/webrtc-app

```
root@sergio:/home/sergio# sudo mkdir -p /var/www/html/webrtc-app
root@sergio:/home/sergio# sudo chown -R $USER:$USER /var/www/html/webrtc-app
root@sergio:/home/sergio# sudo chmod -R 755 /var/www/html/webrtc-app
root@sergio:/home/sergio#
```

1.3.2 Configuration du Virtual Host

Création du fichier de configuration

sudo nano /etc/apache2/sites-available/webrtc-app.conf

```
root@sergio:/home/sergio# sudo nano /etc/apache2/sites-available/webrtc-app.conf
root@sergio:/home/sergio#
```

Contenu du fichier **webrtc-app.conf**

```
<VirtualHost *:80>
    ServerName localhost
    DocumentRoot /var/www/html/webrtc-app

    # Headers nécessaires pour WebRTC
    Header always set Access-Control-Allow-Origin "*"
    Header always set Access-Control-Allow-Methods "GET, POST, OPTIONS, PUT,
DELETE"
    Header always set Access-Control-Allow-Headers "Content-Type,
Authorization, X-Requested-With"

    # Support des WebSockets
    LoadModule proxy_module modules/mod_proxy.so
    LoadModule proxy_http_module modules/mod_proxy_http.so
    LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so

    <Directory "/var/www/html/webrtc-app">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/webrtc_error.log
    CustomLog ${APACHE_LOG_DIR}/webrtc_access.log combined
</VirtualHost>
```

1.3.3 Activation du site et des modules

On active les modules nécessaires par les comandes suivantes

sudo a2enmod headers

sudo a2enmod rewrite

```
root@sergio:/home/sergio# sudo a2enmod headers
Module headers already enabled
root@sergio:/home/sergio# sudo a2enmod rewrite
Module rewrite already enabled
root@sergio:/home/sergio#
```

sudo a2enmod proxy

```
root@sergio:/home/sergio# sudo a2enmod proxy
Enabling module proxy.
To activate the new configuration, you need to run:
```

sudo a2enmod proxy_http

```
root@sergio:/home/sergio# sudo a2enmod proxy_http
Considering dependency proxy for proxy_http:
Module proxy already enabled
Enabling module proxy_http.
```

sudo a2enmod proxy_wstunnel

```
root@sergio:/home/sergio# sudo a2enmod proxy_wstunnel
Considering dependency proxy for proxy_wstunnel:
Module proxy already enabled
Enabling module proxy_wstunnel.
```

systemctl restart apache2

```
root@sergio:/home/sergio# systemctl restart apache2
root@sergio:/home/sergio#
```

1.3.4 Activation du site

sudo a2ensite webrtc-app.conf

```
root@sergio:/home/sergio# sudo a2ensite webrtc-app.conf
Enabling site webrtc-app.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@sergio:/home/sergio#
```

1.3.5 Désactivation du site par défaut (optionnel)

sudo a2dissite 000-default.conf

```
root@sergio:/home/sergio# sudo a2dissite 000-default.conf
Site 000-default already disabled
root@sergio:/home/sergio#
```

1.3.6 Redémarrage d'Apache2

sudo systemctl restart apache2

```
root@sergio:/home/sergio# sudo systemctl restart apache2
root@sergio:/home/sergio#
```

1.4 Structure du Projet

Nous allons créer le dossier comme suit

cd /var/www/html/webrtc-app

```
root@sergio:/home/sergio# cd /var/www/html/webrtc-app
root@sergio:/var/www/html/webrtc-app#
```

Puis on créer la structure du dossier par les commandes suivantes

mkdir -p {css,js,assets}

touch index.html css/style.css js/main.js js/peerjs-config.js

```
root@sergio:/var/www/html/webrtc-app# mkdir -p {css,js,assets}
root@sergio:/var/www/html/webrtc-app# touch index.html css/style.css js/main.js js/peerjs-config.js
root@sergio:/var/www/html/webrtc-app#
```

Voici la structure finale

```
webrtc-app/
├── index.html
├── css/
│   └── style.css
├── js/
│   ├── main.js
│   └── peerjs-config.js
└── assets/
```

1.5 Code de l'Application WebRTC

- Fichier HTML Principal (index.html)

nano index.html

```
root@sergio:/var/www/html/webrtc-app# nano index.html
root@sergio:/var/www/html/webrtc-app#
```

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Application WebRTC - Conférence + Chat + Partage d'Écran</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="container">
        <header>
            <h1>📹 WebRTC Conference App</h1>
            <div class="user-info">
                <span id="user-id">ID: Non connecté</span>
                <button id="copy-id" class="btn-small">Copier ID</button>
            </div>
        </header>

        <main>
            <!-- Section Connexion -->
```

```
            <section class="connection-section">
                <div class="input-group">
                    <input type="text" id="peer-id-input" placeholder="Entrez
l'ID du destinataire">
                    <button id="connect-btn" class="btn-primary">Se
connecter</button>
                </div>
                <div class="controls">
                    <button id="call-btn" class="btn-success"
disabled>Appeler</button>
                    <button id="hang-up-btn" class="btn-danger"
disabled>Raccrocher</button>
                    <button id="share-screen-btn" class="btn-info"
disabled>Partager l'écran</button>
                    <button id="stop-share-btn" class="btn-warning"
disabled>Arrêter partage</button>
                </div>
            </section>

            <!-- Section Vidéo -->
            <section class="video-section">
                <div class="video-container">
                    <div class="video-wrapper">
                        <video id="local-video" autoplay muted
playsinline></video>
                        <label>Votre vidéo</label>
                    </div>
                    <div class="video-wrapper">
                        <video id="remote-video" autoplay playsinline></video>
                        <label>Vidéo distante</label>
                    </div>
                </div>
            </section>

            <!-- Section Chat -->
            <section class="chat-section">
                <div class="chat-container">
                    <div class="chat-header">
                        <h3>💬 Chat</h3>
                        <button id="clear-chat" class="btn-
small">Effacer</button>
                    </div>
                    <div id="chat-messages" class="chat-messages"></div>
                    <div class="chat-input">
                        <input type="text" id="message-input"
placeholder="Tapez votre message...">
                        <button id="send-message" class="btn-
primary">Envoyer</button>
                    </div>
```

```
                </div>
            </section>
        </main>

        <!-- Status et Logs -->
        <footer>
            <div class="status-bar">
                <span id="connection-status">⬤ Déconnecté</span>
                <span id="call-status">📞 Pas d'appel</span>
            </div>
            <div class="logs">
                <h4>Logs de connexion</h4>
                <div id="logs-container"></div>
            </div>
        </footer>
    </div>

    <!-- Scripts -->
    <script src="https://unpkg.com/peerjs@1.4.7/dist/peerjs.min.js"></script>
    <script src="js/peerjs-config.js"></script>
    <script src="js/main.js"></script>
</body>
</html>
```

- Fichier CSS (css/style.css)

nano css/style.css

```
root@sergio:/var/www/html/webrtc-app# nano css/style.css
root@sergio:/var/www/html/webrtc-app#
```

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    color: #333;
}

.container {
    max-width: 1400px;
    margin: 0 auto;
    padding: 20px;
```

```css
}

header {
    background: rgba(255, 255, 255, 0.95);
    border-radius: 15px;
    padding: 20px;
    margin-bottom: 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
}

h1 {
    color: #4a5568;
    font-size: 2rem;
}

.user-info {
    display: flex;
    align-items: center;
    gap: 10px;
}

#user-id {
    background: #e2e8f0;
    padding: 8px 12px;
    border-radius: 8px;
    font-family: monospace;
    font-size: 0.9rem;
}

/* Boutons */
.btn-primary, .btn-success, .btn-danger, .btn-info, .btn-warning, .btn-small {
    padding: 10px 20px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-weight: 600;
    transition: all 0.3s ease;
    text-transform: uppercase;
    font-size: 0.9rem;
}

.btn-small {
    padding: 5px 10px;
    font-size: 0.8rem;
}
```

```css
.btn-primary { background: #4299e1; color: white; }
.btn-success { background: #48bb78; color: white; }
.btn-danger { background: #f56565; color: white; }
.btn-info { background: #38b2ac; color: white; }
.btn-warning { background: #ed8936; color: white; }

.btn-primary:hover { background: #3182ce; }
.btn-success:hover { background: #38a169; }
.btn-danger:hover { background: #e53e3e; }
.btn-info:hover { background: #319795; }
.btn-warning:hover { background: #dd6b20; }

button:disabled {
    opacity: 0.5;
    cursor: not-allowed;
}

/* Sections */
.connection-section, .video-section, .chat-section {
    background: rgba(255, 255, 255, 0.95);
    border-radius: 15px;
    padding: 20px;
    margin-bottom: 20px;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
}

.input-group {
    display: flex;
    gap: 10px;
    margin-bottom: 15px;
}

#peer-id-input, #message-input {
    flex: 1;
    padding: 12px;
    border: 2px solid #e2e8f0;
    border-radius: 8px;
    font-size: 1rem;
}

#peer-id-input:focus, #message-input:focus {
    outline: none;
    border-color: #4299e1;
}

.controls {
    display: flex;
    gap: 10px;
    flex-wrap: wrap;
```

```
}

/* Vidéos */
.video-container {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
    margin: 20px 0;
}

.video-wrapper {
    position: relative;
    background: #000;
    border-radius: 12px;
    overflow: hidden;
    aspect-ratio: 16/9;
}

.video-wrapper label {
    position: absolute;
    bottom: 10px;
    left: 10px;
    background: rgba(0, 0, 0, 0.7);
    color: white;
    padding: 5px 10px;
    border-radius: 5px;
    font-size: 0.8rem;
}

video {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

/* Chat */
.chat-container {
    height: 400px;
    display: flex;
    flex-direction: column;
}

.chat-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 15px;
    padding-bottom: 10px;
    border-bottom: 2px solid #e2e8f0;
```

```
}

.chat-messages {
    flex: 1;
    overflow-y: auto;
    padding: 10px;
    background: #f7fafc;
    border-radius: 8px;
    margin-bottom: 15px;
}

.chat-input {
    display: flex;
    gap: 10px;
}

.message {
    margin-bottom: 10px;
    padding: 8px 12px;
    border-radius: 12px;
    max-width: 80%;
    word-wrap: break-word;
}

.message.sent {
    background: #4299e1;
    color: white;
    margin-left: auto;
    text-align: right;
}

.message.received {
    background: #e2e8f0;
    color: #333;
}

.message-time {
    font-size: 0.7rem;
    opacity: 0.7;
    margin-top: 3px;
}

/* Footer */
footer {
    background: rgba(255, 255, 255, 0.95);
    border-radius: 15px;
    padding: 20px;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
}
```

```css
.status-bar {
    display: flex;
    gap: 20px;
    margin-bottom: 15px;
    font-weight: 600;
}

.logs {
    background: #f7fafc;
    border-radius: 8px;
    padding: 15px;
}

#logs-container {
    max-height: 200px;
    overflow-y: auto;
    font-family: monospace;
    font-size: 0.85rem;
    line-height: 1.4;
}

.log-entry {
    margin-bottom: 5px;
    padding: 2px 0;
}

.log-info { color: #4299e1; }
.log-success { color: #48bb78; }
.log-error { color: #f56565; }
.log-warning { color: #ed8936; }

/* Responsive */
@media (max-width: 768px) {
    .video-container {
        grid-template-columns: 1fr;
    }

    .controls {
        justify-content: center;
    }

    header {
        flex-direction: column;
        gap: 15px;
        text-align: center;
    }

    .input-group {
```

```
        flex-direction: column;
    }
}
```

- Configuration PeerJS (js/peerjs-config.js)

nano js/peerjs-config.js

```
root@sergio:/var/www/html/webrtc-app# nano js/peerjs-config.js
root@sergio:/var/www/html/webrtc-app#
```

```
// Configuration PeerJS
const PEERJS_CONFIG = {
    host: 'localhost',
    port: 9000,
    path: '/myapp',
    debug: 2,
    config: {
        iceServers: [
            { urls: 'stun:stun.l.google.com:19302' },
            { urls: 'stun:stun1.l.google.com:19302' },
            {
                urls: 'turn:your-turn-server.com:3478',
                username: 'your-username',
                credential: 'your-password'
            }
        ]
    }
};

// Configuration alternative pour utiliser le serveur PeerJS public
const PEERJS_CONFIG_PUBLIC = {
    debug: 2,
    config: {
        iceServers: [
            { urls: 'stun:stun.l.google.com:19302' },
            { urls: 'stun:stun1.l.google.com:19302' }
        ]
    }
};
```

- Script Principal (js/main.js)

nano js/main.js

```
root@sergio:/var/www/html/webrtc-app# nano js/main.js
root@sergio:/var/www/html/webrtc-app#
```

```

class WebRTCApp {
```

```
    constructor() {
        this.peer = null;
        this.localStream = null;
        this.remoteStream = null;
        this.currentCall = null;
        this.dataConnection = null;
        this.isScreenSharing = false;
        this.originalStream = null;

        this.initializeElements();
        this.initializePeer();
        this.setupEventListeners();
        this.initializeLocalVideo();
    }

    initializeElements() {
        // Éléments du DOM
        this.elements = {
            userIdDisplay: document.getElementById('user-id'),
            copyIdBtn: document.getElementById('copy-id'),
            peerIdInput: document.getElementById('peer-id-input'),
            connectBtn: document.getElementById('connect-btn'),
            callBtn: document.getElementById('call-btn'),
            hangUpBtn: document.getElementById('hang-up-btn'),
            shareScreenBtn: document.getElementById('share-screen-btn'),
            stopShareBtn: document.getElementById('stop-share-btn'),
            localVideo: document.getElementById('local-video'),
            remoteVideo: document.getElementById('remote-video'),
            chatMessages: document.getElementById('chat-messages'),
            messageInput: document.getElementById('message-input'),
            sendMessageBtn: document.getElementById('send-message'),
            clearChatBtn: document.getElementById('clear-chat'),
            connectionStatus: document.getElementById('connection-status'),
            callStatus: document.getElementById('call-status'),
            logsContainer: document.getElementById('logs-container')
        };
    }

    initializePeer() {
        try {
            // Utilisation de la configuration publique pour la démo
            this.peer = new Peer(PEERJS_CONFIG_PUBLIC);

            this.peer.on('open', (id) => {
                this.log(`Connexion établie avec l'ID: ${id}`, 'success');
                this.elements.userIdDisplay.textContent = `ID: ${id}`;
                this.updateConnectionStatus('🟢 Connecté');
            });
```

```
            this.peer.on('error', (err) => {
                this.log(`Erreur PeerJS: ${err}`, 'error');
                this.updateConnectionStatus('🔴 Erreur de connexion');
            });

            this.peer.on('call', (call) => {
                this.log('Appel entrant reçu', 'info');
                this.handleIncomingCall(call);
            });

            this.peer.on('connection', (conn) => {
                this.log('Connexion de données entrante', 'info');
                this.setupDataConnection(conn);
            });

        } catch (error) {
            this.log(`Erreur d'initialisation: ${error.message}`, 'error');
        }
    }

    setupEventListeners() {
        // Boutons de connexion
        this.elements.copyIdBtn.addEventListener('click', () =>
this.copyUserId());
        this.elements.connectBtn.addEventListener('click', () =>
this.connectToPeer());
        this.elements.callBtn.addEventListener('click', () =>
this.startCall());
        this.elements.hangUpBtn.addEventListener('click', () =>
this.hangUp());

        // Partage d'écran
        this.elements.shareScreenBtn.addEventListener('click', () =>
this.startScreenShare());
        this.elements.stopShareBtn.addEventListener('click', () =>
this.stopScreenShare());

        // Chat
        this.elements.sendMessageBtn.addEventListener('click', () =>
this.sendMessage());
        this.elements.messageInput.addEventListener('keypress', (e) => {
            if (e.key === 'Enter') this.sendMessage();
        });
        this.elements.clearChatBtn.addEventListener('click', () =>
this.clearChat());

        // Entrée clavier pour connexion
        this.elements.peerIdInput.addEventListener('keypress', (e) => {
            if (e.key === 'Enter') this.connectToPeer();
```

```
        });
    }

    async initializeLocalVideo() {
        try {
            this.localStream = await navigator.mediaDevices.getUserMedia({
                video: { width: 1280, height: 720 },
                audio: true
            });

            this.elements.localVideo.srcObject = this.localStream;
            this.originalStream = this.localStream;
            this.log('Caméra et microphone initialisés', 'success');

        } catch (error) {
            this.log(`Erreur d'accès aux médias: ${error.message}`, 'error');
        }
    }

    copyUserId() {
        if (this.peer && this.peer.id) {
            navigator.clipboard.writeText(this.peer.id).then(() => {
                this.log('ID copié dans le presse-papiers', 'success');
            });
        }
    }

    connectToPeer() {
        const peerId = this.elements.peerIdInput.value.trim();
        if (!peerId) {
            this.log('Veuillez entrer un ID de destinataire', 'warning');
            return;
        }

        this.log(`Tentative de connexion à: ${peerId}`, 'info');

        // Établir une connexion de données
        this.dataConnection = this.peer.connect(peerId);
        this.setupDataConnection(this.dataConnection);

        // Activer les boutons
        this.elements.callBtn.disabled = false;
        this.elements.shareScreenBtn.disabled = false;
    }

    setupDataConnection(conn) {
        this.dataConnection = conn;

        conn.on('open', () => {
```

```
            this.log('Connexion de données établie', 'success');
            this.elements.callBtn.disabled = false;
            this.elements.shareScreenBtn.disabled = false;
        });

        conn.on('data', (data) => {
            if (data.type === 'chat') {
                this.displayMessage(data.message, false);
            }
        });

        conn.on('close', () => {
            this.log('Connexion de données fermée', 'warning');
        });

        conn.on('error', (err) => {
            this.log(`Erreur de connexion de données: ${err}`, 'error');
        });
    }

    startCall() {
        if (!this.localStream) {
            this.log('Stream local non disponible', 'error');
            return;
        }

        const peerId = this.elements.peerIdInput.value.trim();
        this.log(`Démarrage de l'appel vers: ${peerId}`, 'info');

        this.currentCall = this.peer.call(peerId, this.localStream);
        this.setupCall(this.currentCall);
    }

    handleIncomingCall(call) {
        this.log('Réception d\'un appel entrant', 'info');

        if (confirm('Voulez-vous accepter cet appel ?')) {
            call.answer(this.localStream);
            this.currentCall = call;
            this.setupCall(call);
        } else {
            call.close();
        }
    }

    setupCall(call) {
        call.on('stream', (remoteStream) => {
            this.log('Stream distant reçu', 'success');
            this.elements.remoteVideo.srcObject = remoteStream;
```

```
            this.updateCallStatus('📞 En appel');
            this.elements.hangUpBtn.disabled = false;
        });

        call.on('close', () => {
            this.log('Appel terminé', 'info');
            this.elements.remoteVideo.srcObject = null;
            this.updateCallStatus('📞 Pas d\'appel');
            this.elements.hangUpBtn.disabled = true;
        });

        call.on('error', (err) => {
            this.log(`Erreur d'appel: ${err}`, 'error');
        });
    }

    hangUp() {
        if (this.currentCall) {
            this.currentCall.close();
            this.currentCall = null;
        }

        this.elements.remoteVideo.srcObject = null;
        this.updateCallStatus('📞 Pas d\'appel');
        this.elements.hangUpBtn.disabled = true;
        this.log('Appel terminé', 'info');
    }

    async startScreenShare() {
        try {
            const screenStream = await
navigator.mediaDevices.getDisplayMedia({
                video: true,
                audio: true
            });

            // Remplacer le stream vidéo
            if (this.currentCall) {
                const videoTrack = screenStream.getVideoTracks()[0];
                const sender =
this.currentCall.peerConnection.getSenders().find(s =>
                    s.track && s.track.kind === 'video'
                );

                if (sender) {
                    await sender.replaceTrack(videoTrack);
                }
            }
```

```
            this.elements.localVideo.srcObject = screenStream;
            this.isScreenSharing = true;
            this.elements.shareScreenBtn.disabled = true;
            this.elements.stopShareBtn.disabled = false;

            this.log('Partage d\'écran démarré', 'success');

            // Arrêter le partage quand l'utilisateur ferme la fenêtre de
partage
            screenStream.getVideoTracks()[0].onended = () => {
                this.stopScreenShare();
            };

        } catch (error) {
            this.log(`Erreur de partage d'écran: ${error.message}`, 'error');
        }
    }

    async stopScreenShare() {
        try {
            // Revenir à la caméra
            if (this.currentCall && this.originalStream) {
                const videoTrack = this.originalStream.getVideoTracks()[0];
                const sender =
this.currentCall.peerConnection.getSenders().find(s =>
                    s.track && s.track.kind === 'video'
                );

                if (sender) {
                    await sender.replaceTrack(videoTrack);
                }
            }

            this.elements.localVideo.srcObject = this.originalStream;
            this.isScreenSharing = false;
            this.elements.shareScreenBtn.disabled = false;
            this.elements.stopShareBtn.disabled = true;

            this.log('Partage d\'écran arrêté', 'info');

        } catch (error) {
            this.log(`Erreur d'arrêt du partage: ${error.message}`, 'error');
        }
    }

    sendMessage() {
        const message = this.elements.messageInput.value.trim();
        if (!message || !this.dataConnection) return;
```

```
        this.dataConnection.send({
            type: 'chat',
            message: message
        });

        this.displayMessage(message, true);
        this.elements.messageInput.value = '';
    }

    displayMessage(message, isSent) {
        const messageDiv = document.createElement('div');
        messageDiv.className = `message ${isSent ? 'sent' : 'received'}`;

        const time = new Date().toLocaleTimeString();
        messageDiv.innerHTML = `
            <div>${message}</div>
            <div class="message-time">${time}</div>
        `;

        this.elements.chatMessages.appendChild(messageDiv);
        this.elements.chatMessages.scrollTop =
this.elements.chatMessages.scrollHeight;
    }

    clearChat() {
        this.elements.chatMessages.innerHTML = '';
        this.log('Chat effacé', 'info');
    }

    updateConnectionStatus(status) {
        this.elements.connectionStatus.textContent = status;
    }

    updateCallStatus(status) {
        this.elements.callStatus.textContent = status;
    }

    log(message, type = 'info') {
        const timestamp = new Date().toLocaleTimeString();
        const logEntry = document.createElement('div');
        logEntry.className = `log-entry log-${type}`;
        logEntry.textContent = `[${timestamp}] ${message}`;

        this.elements.logsContainer.appendChild(logEntry);
        this.elements.logsContainer.scrollTop =
this.elements.logsContainer.scrollHeight;

        console.log(`[WebRTC] ${message}`);
    }
```

```
}

// Initialisation de l'application
document.addEventListener('DOMContentLoaded', () => {
    new WebRTCApp();
});
```

1.6 Installation et Configuration d'un Serveur PeerJS Local

1.6.1 Installation de PeerJS Server

sudo npm install -g peer

```
root@sergio:/var/www/html/webrtc-app# sudo npm install -g peer
npm warn deprecated node-domexception@1.0.0: Use your platform's native DOMException instead

added 108 packages in 18s

20 packages are looking for funding
  run `npm fund` for details
root@sergio:/var/www/html/webrtc-app#
```

1.6.2 Démarrage du serveur PeerJS

1.6.2.1 Démarrage simple

peerjs --port 9000 --key peerjs --path /myapp

```
root@sergio:/var/www/html/webrtc-app# peerjs --port 9000 --key peerjs --path /myapp
Started PeerServer on ::, port: 9000, path: /myapp
```

1.6.2.2 Démarrage avec configuration avancée

peerjs --port 9000 --key peerjs --path /myapp --allow_discovery true

```
root@sergio:/var/www/html/webrtc-app# peerjs --port 9000 --key peerjs --path /myapp --allow_discovery true
Started PeerServer on ::, port: 9000, path: /myapp
```

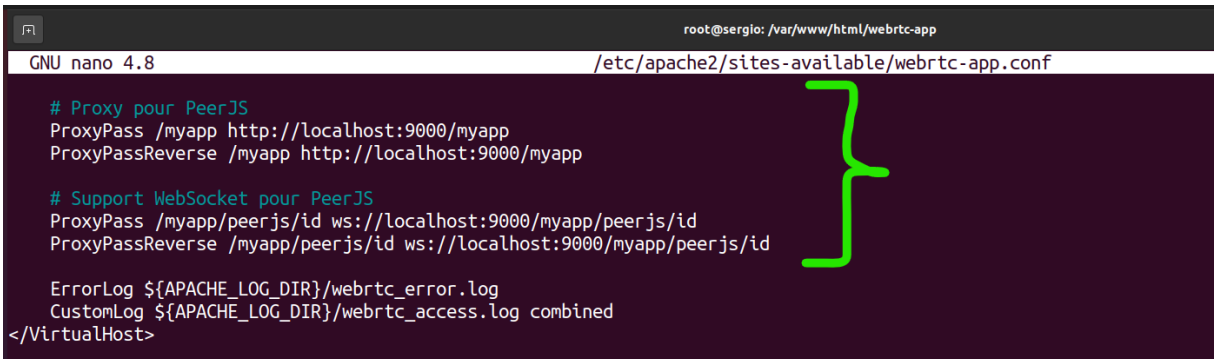
1.6.2.3 Configuration Apache pour proxy PeerJS

Éditons le fichier de la configuration Apache

sudo nano /etc/apache2/sites-available/webrtc-app.conf

```
root@sergio:/var/www/html/webrtc-app# sudo nano /etc/apache2/sites-available/webrtc-app.conf
root@sergio:/var/www/html/webrtc-app#
```

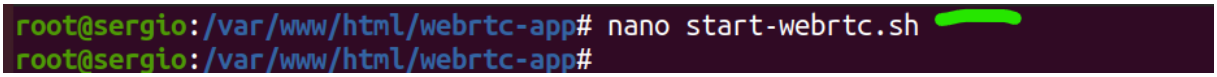
```
# Proxy pour PeerJS
ProxyPass /myapp http://localhost:9000/myapp
ProxyPassReverse /myapp http://localhost:9000/myapp

# Support WebSocket pour PeerJS
ProxyPass /myapp/peerjs/id ws://localhost:9000/myapp/peerjs/id
ProxyPassReverse /myapp/peerjs/id ws://localhost:9000/myapp/peerjs/id
```



1.7 Scripts de Test et de Déploiement

1.7.1 Script de démarrage automatique (start-webrtc.sh)

nano start-webrtc.sh




```
#!/bin/bash

echo "=== Démarrage de l'application WebRTC ==="

# Vérification d'Apache2
if systemctl is-active --quiet apache2; then
    echo "✅ Apache2 est actif"
else
    echo "❌ Démarrage d'Apache2..."
    sudo systemctl start apache2
fi

# Vérification de PeerJS (si installé)
if command -v peerjs &> /dev/null; then
    echo "🔄 Démarrage du serveur PeerJS..."
    peerjs --port 9000 --key peerjs --path /myapp &
    PEERJS_PID=$!
    echo "✅ Serveur PeerJS démarré (PID: $PEERJS_PID)"
fi

# Ouverture du navigateur
sleep 2
echo "🌐 Ouverture de l'application..."
xdg-open http://localhost/webrtc-app 2>/dev/null || open
http://localhost/webrtc-app 2>/dev/null

echo "=== Application WebRTC démarrée ==="
echo "URL: http://localhost/webrtc-app"
```

1.7.2 Script de test de connectivité (test-webrtc.sh)

nano test-webrtc.sh

```
root@sergio:/var/www/html/webrtc-app# nano test-webrtc.sh
root@sergio:/var/www/html/webrtc-app#
```

```
#!/bin/bash

echo "=== Test de l'application WebRTC ==="

# Test Apache2
echo "🔍 Test d'Apache2..."
if curl -s http://localhost/webrtc-app > /dev/null; then
    echo "✅ Apache2 et application accessible"
else
    echo "❌ Problème d'accès à l'application"
    exit 1
fi

# Test PeerJS (si configuré)
echo "🔍 Test du serveur PeerJS..."
if curl -s http://localhost:9000/myapp > /dev/null; then
    echo "✅ Serveur PeerJS accessible"
else
    echo "⚠️  Serveur PeerJS non accessible (normal si non configuré)"
fi

# Test des permissions de fichiers
echo "🔍 Test des permissions..."
if [ -r "/var/www/html/webrtc-app/index.html" ]; then
    echo "✅ Fichiers accessibles"
else
    echo "❌ Problème de permissions des fichiers"
fi

echo "=== Test terminé ==="
```

sudo chown -R www-data:www-data /var/www/html/webrtc-app/

```
root@sergio:/var/www/html/webrtc-app# sudo chown -R www-data:www-data /var/www/html/webrtc-app/
root@sergio:/var/www/html/webrtc-app#
```

sudo chmod -R 755 /var/www/html/webrtc-app/

sudo chmod 644 /var/www/html/webrtc-app/index.html

```
root@sergio:/var/www/html/webrtc-app# sudo chmod -R 755 /var/www/html/webrtc-app/
root@sergio:/var/www/html/webrtc-app# sudo chmod 644 /var/www/html/webrtc-app/index.html
root@sergio:/var/www/html/webrtc-app#
```

1.8 Tests de l'Application

On rend les scripts exécutables

chmod +x start-webrtc.sh test-webrtc.sh

```
root@sergio:/var/www/html/webrtc-app# chmod +x start-webrtc.sh test-webrtc.sh
root@sergio:/var/www/html/webrtc-app#
```

peerjs --port 9000 --key peerjs --path /myapp

```
root@sergio:/var/www/html/webrtc-app# peerjs --port 9000 --key peerjs --path /myapp
Started PeerServer on ::, port: 9000, path: /myapp
```

Lancer les tests

./test-webrtc.sh

```
root@sergio:/var/www/html/webrtc-app# ./test-webrtc.sh
=== Test de l'application WebRTC ===
🔍 Test d'Apache2...
✅ Apache2 et application accessible
🔍 Test du serveur PeerJS...
✅ Serveur PeerJS accessible
🔍 Test des permissions...
✅ Fichiers accessibles
=== Test terminé ===
root@sergio:/var/www/html/webrtc-app#
```

- **Test local**

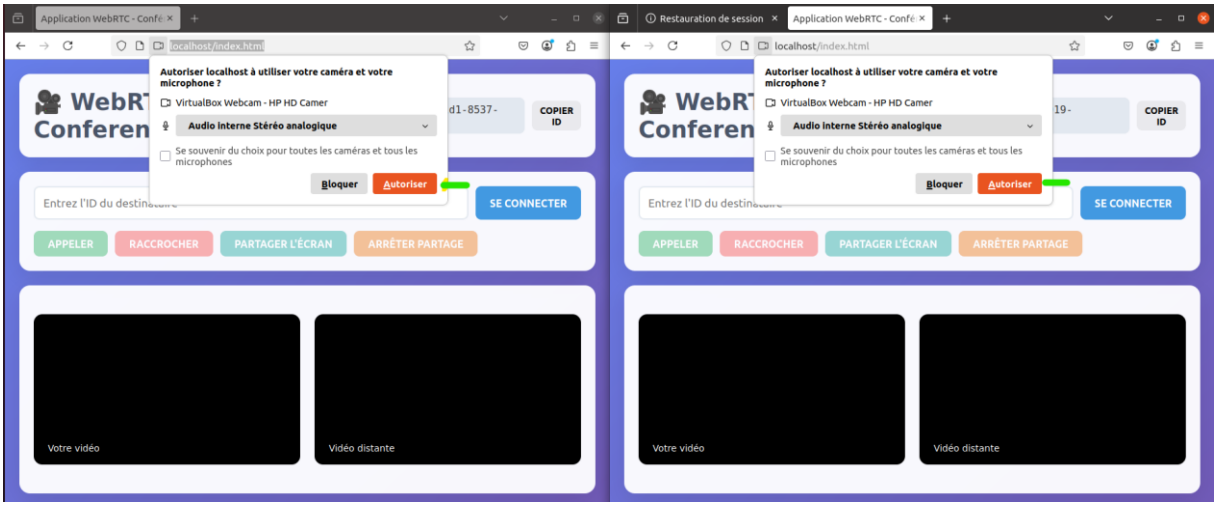
Arrêter d'abord le server peerjs avant d'exécuter la commande suivante

./start-webrtc.sh

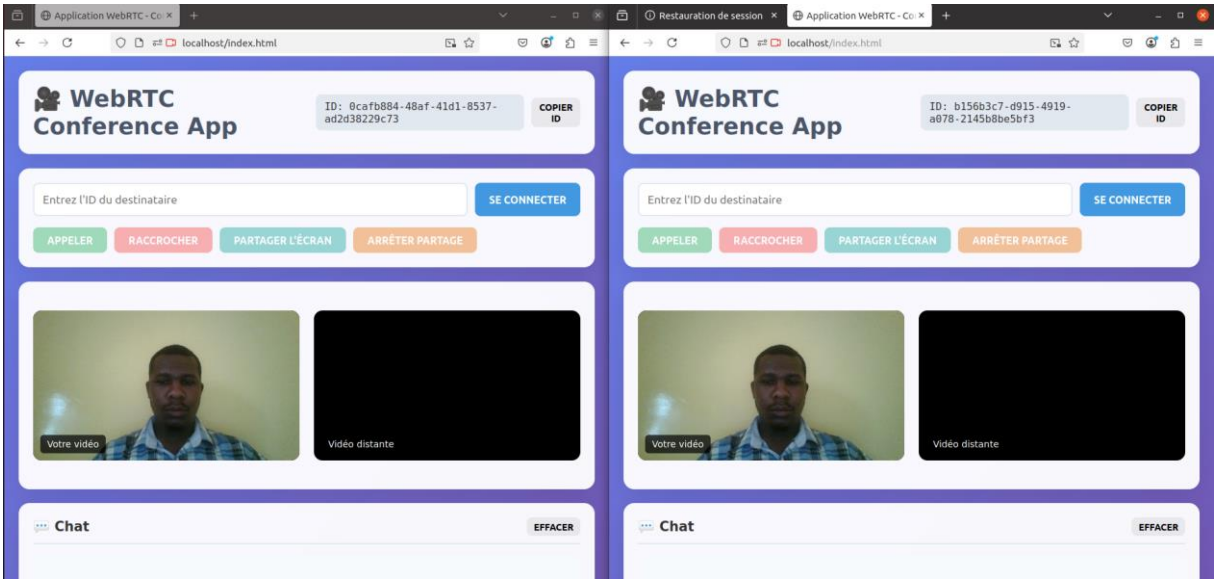
```
root@sergio:/var/www/html/webrtc-app# ./start-webrtc.sh
=== Démarrage de l'application WebRTC ===
✅ Apache2 est actif
🔄 Démarrage du serveur PeerJS...
✅ Serveur PeerJS démarré (PID: 7316)
Started PeerServer on ::, port: 9000, path: /myapp
🌐 Ouverture de l'application...
=== Application WebRTC démarrée ===
URL: http://localhost/webrtc-app
root@sergio:/var/www/html/webrtc-app#
```

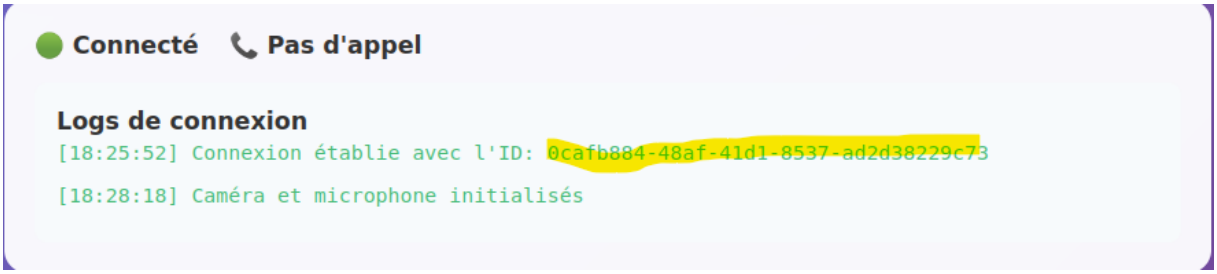
Le lien : http://localhost/index.html

Sur les deux fenetres

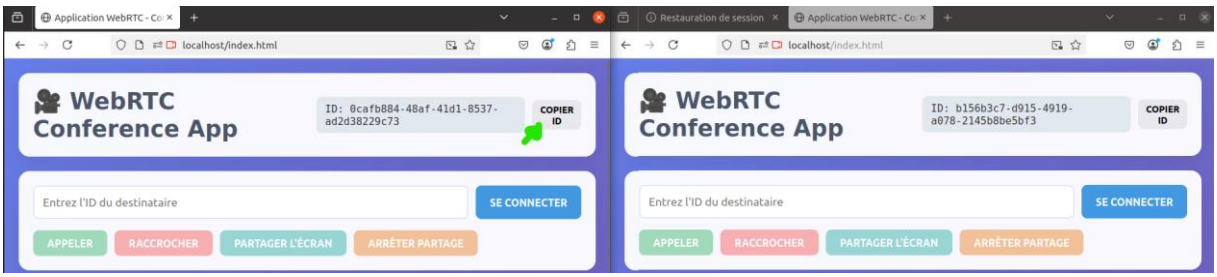


Une fois que l'autorisation est accordée on voie les flux locaux affichés

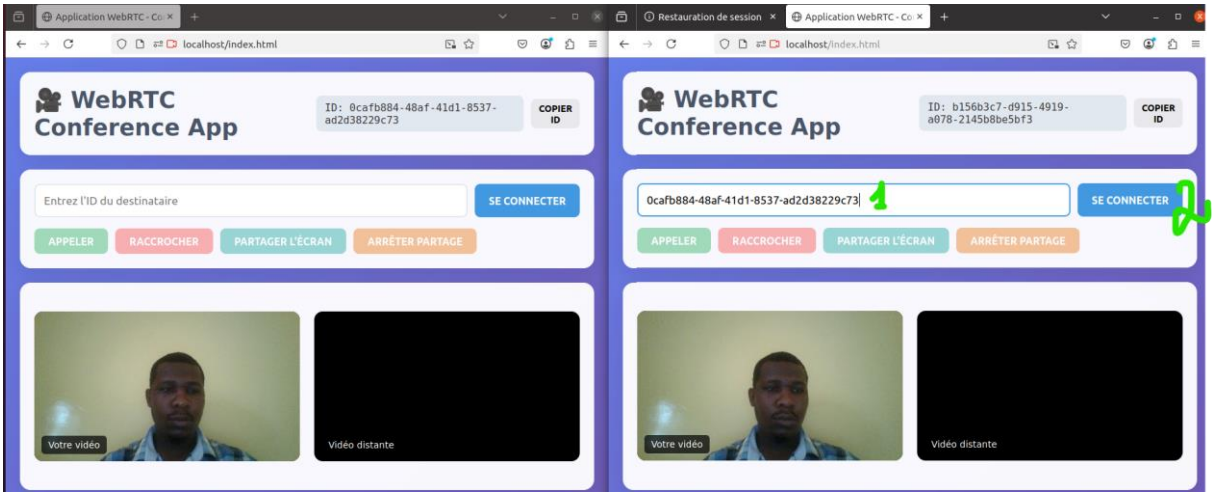


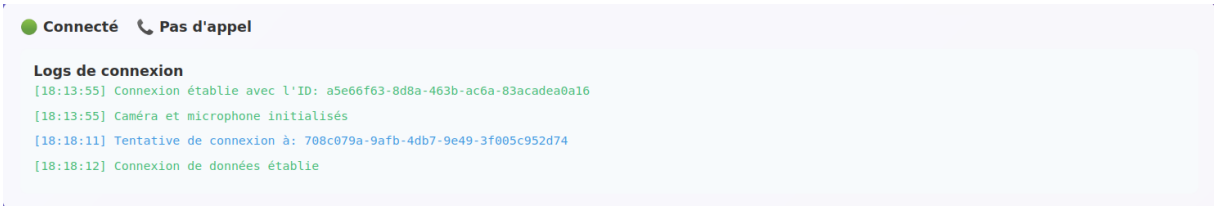


On copie l'ID du fenêtre1 pour coller dans la fenêtre 2 afin de se connecter

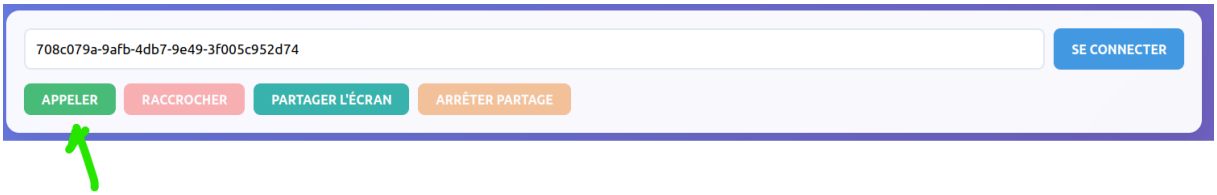


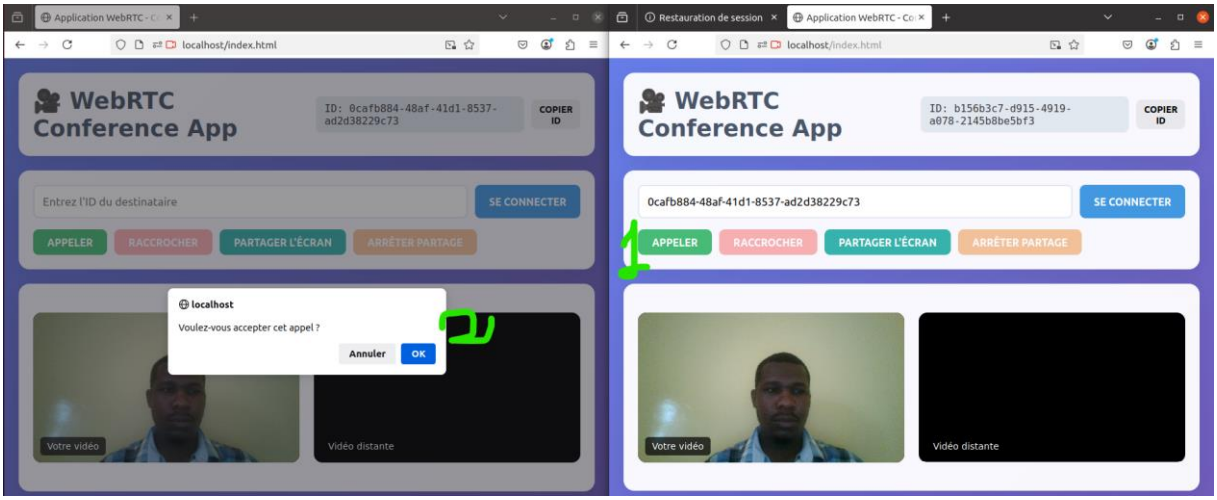
On colle puis on clique sur se connecter et appeler

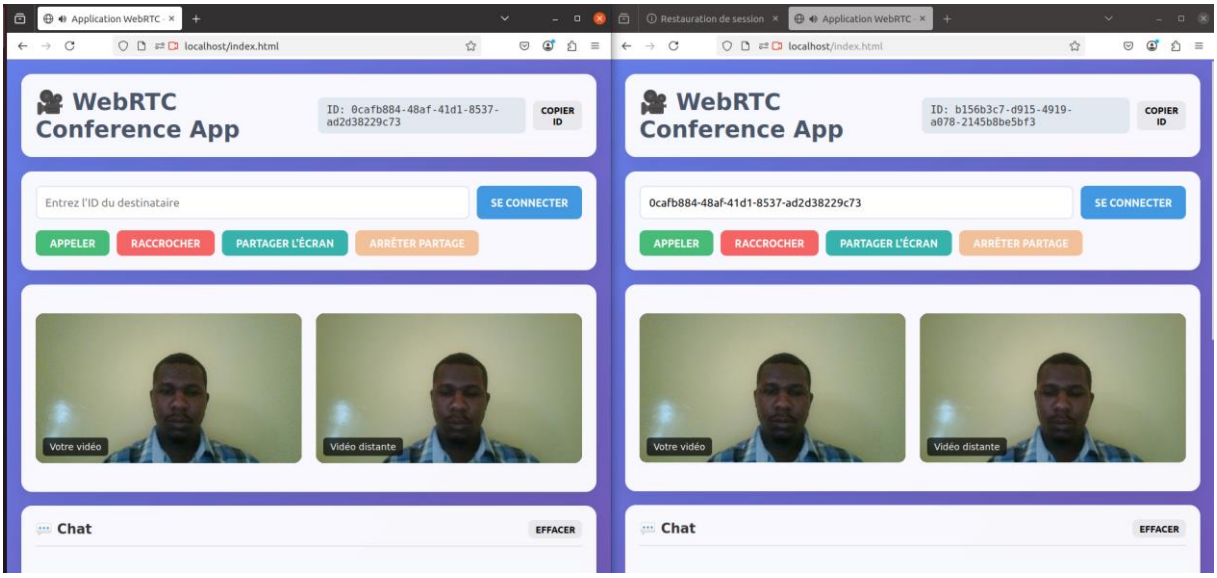


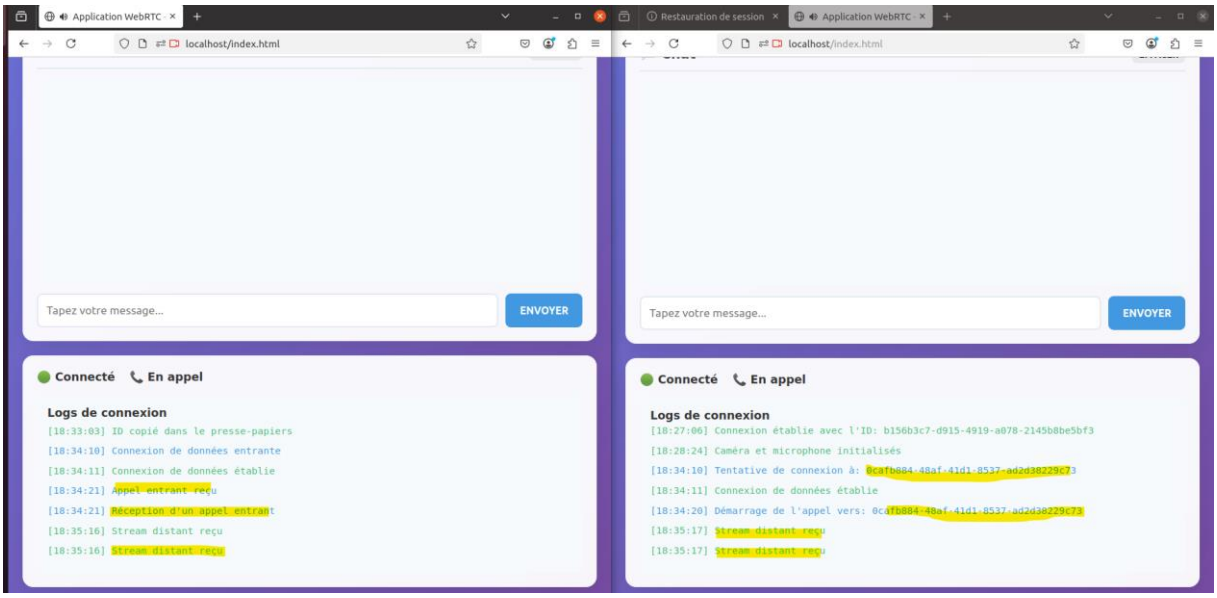


Une foi connecter on clique sur le bouton APPELER

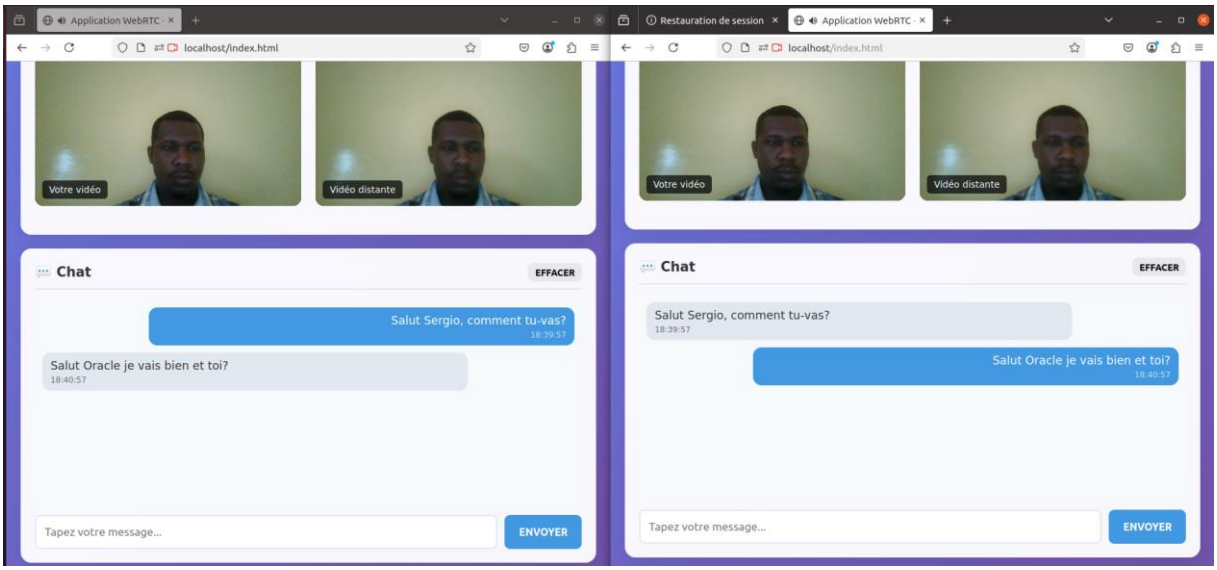




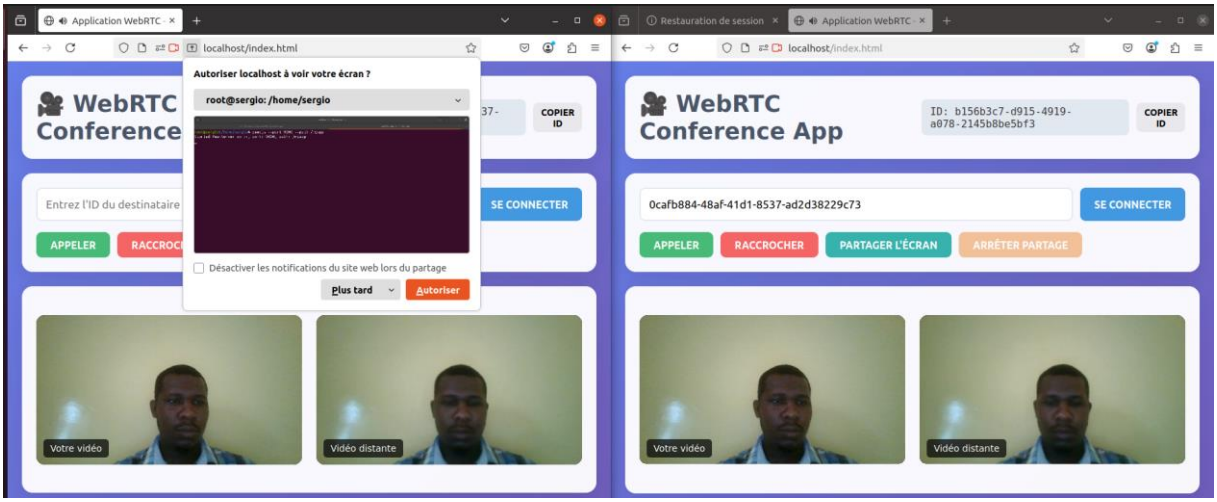


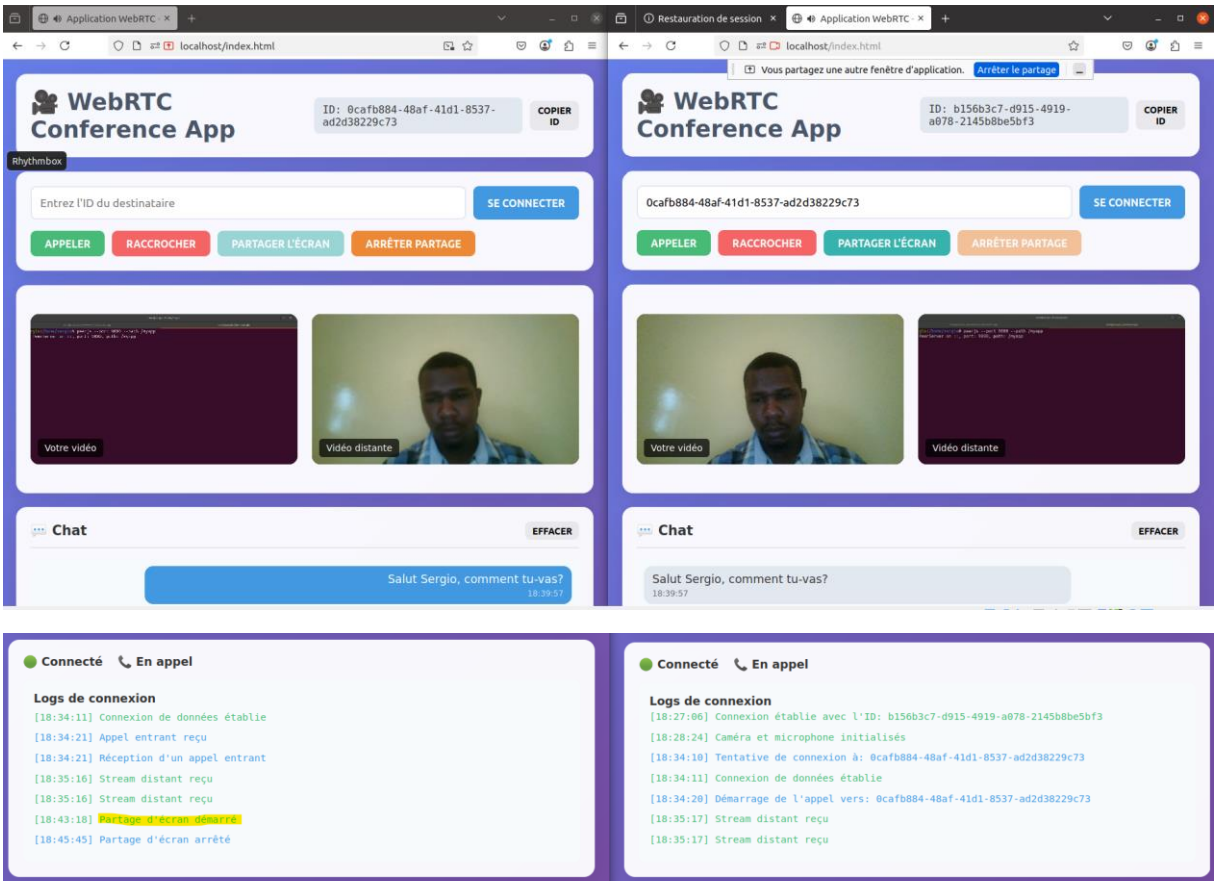


- Chat en directe

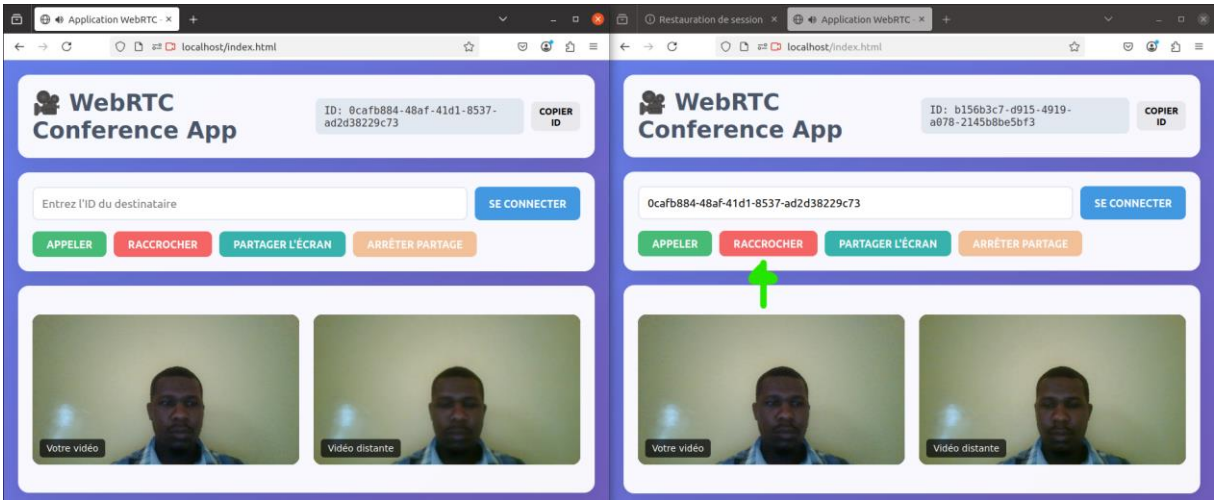


- Partage d'écran





Raccrocher l'appel



2. PeerJS (Serveur de Signalisation avec et sans Express)

2.1 Installation et Configuration de PeerJS Local

2.1.1 Installation de PeerJS Server

Installons globalement le PeerJS par la commande suivante

sudo npm install -g peer

```
root@sergio:/home/sergio# sudo npm install -g peer
npm warn deprecated node-domexception@1.0.0: Use your platform's native DOMException instead

changed 108 packages in 35s

20 packages are looking for funding
  run `npm fund` for details
root@sergio:/home/sergio#
```

2.1.2 Installation locale pour développement (optionnel)

npm install peerjs

```
root@sergio:/home/sergio# npm install peerjs

added 6 packages in 6s

2 packages are looking for funding
  run `npm fund` for details
root@sergio:/home/sergio#
```

2.1.3 Configuration et Test de Base

Créons un dossier pour notre serveur PeerJS

mkdir -p peerjs-server

```
root@sergio:/home/sergio# mkdir peerjs-server
root@sergio:/home/sergio# cd peerjs-server/
root@sergio:/home/sergio/peerjs-server#
```

2.1.4 Démarrage rapide du serveur PeerJS

peerjs --port 9000 --key myapp --path /peerjs

```
root@sergio:/home/sergio/peerjs-server# peerjs --port 9000 --key myapp --path /peerjs
Started PeerServer on ::, port: 9000, path: /peerjs
```

Test dans un autre terminal

curl http://localhost:9000/peerjs

```
root@sergio:/home/sergio# curl http://localhost:9000/peerjs
{"name":"PeerJS Server","description":"A server side element to broker connections between PeerJS clients.","website":"htt
ps://peerjs.com/"}root@sergio:/home/sergio#
root@sergio:/home/sergio#
```

2.2. Serveur de Signalisation Basique SANS Express

2.2.1 Structure du Projet

- Créer la structure du projet

mkdir -p peerjs-basic-server/{config,logs,scripts}

cd peerjs-basic-server/

```
root@sergio:/home/sergio# mkdir -p peerjs-basic-server/{config,logs,scripts}
root@sergio:/home/sergio# cd peerjs-basic-server/
root@sergio:/home/sergio/peerjs-basic-server#
```

2.2.2 Initialiser le projet Node.js

npm init -y

```
root@sergio:/home/sergio/peerjs-basic-server# npm init -y
Wrote to /home/sergio/peerjs-basic-server/package.json:

{
  "name": "peerjs-basic-server",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

2.2.3 Installation des dépendances de base

npm install peer ws uuid

```
root@sergio:/home/sergio/peerjs-basic-server# npm install peer ws uuid
npm warn deprecated node-domexception@1.0.0: Use your platform's native DOMException instead

added 109 packages, and audited 110 packages in 25s

21 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@sergio:/home/sergio/peerjs-basic-server#
```

npm install --save-dev nodemon

```
root@sergio:/home/sergio/peerjs-basic-server# npm install --save-dev nodemon

added 29 packages, and audited 139 packages in 7s

25 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@sergio:/home/sergio/peerjs-basic-server#
```

- Configuration du Serveur Basique (server-basic.js)

nano server-basic.js

```
root@sergio:/home/sergio/peerjs-basic-server# nano server-basic.js
root@sergio:/home/sergio/peerjs-basic-server#
```

```
const { PeerServer } = require('peer');

const fs = require('fs');
const path = require('path');

// Configuration du serveur
const CONFIG = {
    port: 9000,
    path: '/peerjs',
    key: 'webrtc-app',
    allow_discovery: true,
    proxied: false,
    cleanup_out_msgs: 1000,
    concurrent_limit: 5000,
    alive_timeout: 60000,
    expire_timeout: 5000
};

// Création du dossier de logs
const logsDir = path.join(__dirname, 'logs');
if (!fs.existsSync(logsDir)) {
    fs.mkdirSync(logsDir, { recursive: true });
}

// Fonction de logging
function logMessage(type, message, data = null) {
    const timestamp = new Date().toISOString();
    const logEntry = {
        timestamp,
        type,
        message,
        data
    };

    // Log vers la console
    console.log(`[${timestamp}] [${type.toUpperCase()}] ${message}`);
    if (data) console.log('Data:', data);

    // Log vers fichier
    const logFile = path.join(logsDir, `peerjs-${new
Date().toISOString().split('T')[0]}.log`);
    fs.appendFileSync(logFile, JSON.stringify(logEntry) + '\n');
}
```

```
// Création du serveur PeerJS
const server = PeerServer(CONFIG);

// Statistiques du serveur
const stats = {
    totalConnections: 0,
    currentConnections: 0,
    messagesHandled: 0,
    errors: 0,
    startTime: new Date()
};

// Événements du serveur
server.on('connection', (client) => {
    stats.totalConnections++;
    stats.currentConnections++;

    logMessage('connection', `Nouveau client connecté`, {
        clientId: client.getId(),
        token: client.getToken(),
        totalConnections: stats.totalConnections,
        currentConnections: stats.currentConnections
    });

    // Événements du client
    client.on('disconnect', () => {
        stats.currentConnections--;
        logMessage('disconnect', `Client déconnecté`, {
            clientId: client.getId(),
            currentConnections: stats.currentConnections
        });
    });

    client.on('error', (error) => {
        stats.errors++;
        logMessage('error', `Erreur client`, {
            clientId: client.getId(),
            error: error.message
        });
    });
});

server.on('disconnect', (client) => {
    logMessage('server_disconnect', `Déconnexion serveur`, {
        clientId: client.getId()
    });
});
```

```
server.on('message', (client, message) => {
    stats.messagesHandled++;
    logMessage('message', `Message reçu`, {
        clientId: client.getId(),
        messageType: message.type,
        messagesHandled: stats.messagesHandled
    });
});

// Démarrage du serveur
server.listen(() => {
    logMessage('server_start', `Serveur PeerJS démarré`, {
        port: CONFIG.port,
        path: CONFIG.path,
        key: CONFIG.key,
        pid: process.pid
    });

    console.log('\n🚀 Serveur PeerJS Basique Démarré !');
    console.log(`📡 URL: http://localhost:${CONFIG.port}${CONFIG.path}`);
    console.log(`🔑 Clé: ${CONFIG.key}`);
    console.log(`📁 Logs: ${logsDir}`);
    console.log('\n✋ Appuyez sur CTRL+C pour arrêter\n');
});

// Gestion des signaux système
process.on('SIGINT', () => {
    logMessage('server_stop', `Arrêt du serveur demandé`, {
        uptime: Date.now() - stats.startTime.getTime(),
        totalConnections: stats.totalConnections,
        messagesHandled: stats.messagesHandled
    });

    console.log('\n🛑 Arrêt du serveur...');
    process.exit(0);
});

process.on('SIGTERM', () => {
    logMessage('server_terminate', `Serveur terminé`, stats);
    process.exit(0);
});

// API basique pour les statistiques (sans Express)
const http = require('http');
const url = require('url');

// Serveur HTTP basique pour les stats
const statsServer = http.createServer((req, res) => {
    const parsedUrl = url.parse(req.url, true);
```

```
    // Headers CORS
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS');
    res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
    res.setHeader('Content-Type', 'application/json');

    if (req.method === 'OPTIONS') {
        res.writeHead(200);
        res.end();
        return;
    }

    // Routes basiques
    switch (parsedUrl.pathname) {
        case '/stats':
            const serverStats = {
                ...stats,
                uptime: Date.now() - stats.startTime.getTime(),
                uptimeFormatted: formatUptime(Date.now() -
stats.startTime.getTime())
            };
            res.writeHead(200);
            res.end(JSON.stringify(serverStats, null, 2));
            break;

        case '/health':
            res.writeHead(200);
            res.end(JSON.stringify({ status: 'healthy', timestamp: new
Date().toISOString() }));
            break;

        case '/clients':
            // Note: Cette implémentation est basique,
            // pour une vraie liste il faudrait stocker les clients
            res.writeHead(200);
            res.end(JSON.stringify({
                currentConnections: stats.currentConnections,
                message: 'Liste détaillée nécessite Express'
            }));
            break;

        default:
            res.writeHead(404);
            res.end(JSON.stringify({ error: 'Route non trouvée' }));
    }
});

// Démarrer le serveur de stats sur un port différent
```

```
const STATS_PORT = 9001;
statsServer.listen(STATS_PORT, () => {
    logMessage('stats_server', `Serveur de statistiques démarré sur le port
${STATS_PORT}`);
    console.log(`📊 Stats API: http://localhost:${STATS_PORT}/stats`);
});

// Fonction utilitaire pour formater l'uptime
function formatUptime(milliseconds) {
    const seconds = Math.floor(milliseconds / 1000);
    const minutes = Math.floor(seconds / 60);
    const hours = Math.floor(minutes / 60);
    const days = Math.floor(hours / 24);

    if (days > 0) return `${days}j ${hours % 24}h ${minutes % 60}m`;
    if (hours > 0) return `${hours}h ${minutes % 60}m ${seconds % 60}s`;
    if (minutes > 0) return `${minutes}m ${seconds % 60}s`;
    return `${seconds}s`;
}

// Affichage périodique des stats (toutes les 30 secondes)
setInterval(() => {
    logMessage('stats_periodic', `Statistiques périodiques`, {
        ...stats,
        uptime: formatUptime(Date.now() - stats.startTime.getTime())
    });
}, 30000);

module.exports = server;
```

- Script de Configuration (config/server-config.js)

nano config/server-config.js

```
root@sergio:/home/sergio/peerjs-basic-server# nano config/server-config.js
root@sergio:/home/sergio/peerjs-basic-server#
```

```
// Configuration centralisée du serveur PeerJS
module.exports = {
    // Configuration du serveur principal
    server: {
        port: parseInt(process.env.PEERJS_PORT) || 9000,
        path: process.env.PEERJS_PATH || '/peerjs',
        key: process.env.PEERJS_KEY || 'webrtc-app',
        host: process.env.PEERJS_HOST || 'localhost',
        proxied: process.env.PEERJS_PROXIED === 'true' || false,
        allow_discovery: process.env.PEERJS_DISCOVERY === 'true' || true
    },
```

```
    // Configuration des limites
    limits: {
        concurrent_limit: parseInt(process.env.CONCURRENT_LIMIT) || 5000,
        alive_timeout: parseInt(process.env.ALIVE_TIMEOUT) || 60000,
        expire_timeout: parseInt(process.env.EXPIRE_TIMEOUT) || 5000,
        cleanup_out_msgs: parseInt(process.env.CLEANUP_MSGS) || 1000
    },

    // Configuration des logs
    logging: {
        enabled: process.env.LOGGING_ENABLED !== 'false',
        level: process.env.LOG_LEVEL || 'info',
        file: process.env.LOG_FILE || true,
        console: process.env.LOG_CONSOLE !== 'false'
    },

    // Configuration des statistiques
    stats: {
        enabled: process.env.STATS_ENABLED !== 'false',
        port: parseInt(process.env.STATS_PORT) || 9001,
        interval: parseInt(process.env.STATS_INTERVAL) || 30000
    }
};
```

3. Serveur de Signalisation AVEC Express

3.1 Installation des Dépendances Express

npm install express cors helmet morgan express-rate-limit

```
root@sergio:/home/sergio/peerjs-basic-server# npm install express cors helmet morgan express-rate-limit

added 37 packages, removed 4 packages, changed 22 packages, and audited 172 packages in 35s

29 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

npm install --save-dev jest supertest

```
root@sergio:/home/sergio/peerjs-basic-server# npm install --save-dev jest supertest
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if
you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported

added 269 packages, and audited 441 packages in 4m

74 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

nano server-express.js

```
root@sergio:/home/sergio/peerjs-basic-server# nano server-express.js
root@sergio:/home/sergio/peerjs-basic-server#
```

```
#!/usr/bin/env node
```

```
const express = require('express');
const { PeerServer } = require('peer');
const cors = require('cors');
const path = require('path');

console.log('🚀 Démarrage du serveur Express...');

// Configuration
const EXPRESS_PORT = 9002;
const PEERJS_PATH = '/peerjs';

// Créer l'application Express
const app = express();

// Middlewares
app.use(cors({
    origin: '*',
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
    allowedHeaders: ['Content-Type', 'Authorization']
}));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Statistiques
const stats = {
    totalConnections: 0,
    currentConnections: 0,
    startTime: new Date(),
    messagesHandled: 0
};

// Stockage des clients
const connectedClients = new Map();

// ===== CRÉATION DU SERVEUR PEERJS INTÉGRÉ =====
// IMPORTANT: Ne pas spécifier de port ici, il sera intégré dans Express
const peerServer = PeerServer({
    path: PEERJS_PATH,
    key: 'webrtc-app-express',
    allow_discovery: true,
    debug: 1
});

// Intégrer PeerJS dans Express (CRITIQUE: pas de port spécifié)
app.use(PEERJS_PATH, peerServer);

// ===== ROUTES API =====
```

```
// Page d'accueil avec dashboard
app.get('/', (req, res) => {
    res.send(`
    <!DOCTYPE html>
    <html>
    <head>
        <title>Serveur PeerJS Express</title>
        <meta charset="UTF-8">
        <style>
            body {
                font-family: Arial, sans-serif;
                max-width: 1000px;
                margin: 50px auto;
                padding: 20px;
                background: #f5f5f5;
            }
            .container {
                background: white;
                padding: 30px;
                border-radius: 10px;
                box-shadow: 0 4px 6px rgba(0,0,0,0.1);
            }
            .stats {
                background: #e3f2fd;
                padding: 15px;
                border-radius: 8px;
                margin: 15px 0;
                border-left: 4px solid #2196f3;
            }
            button {
                background: #2196f3;
                color: white;
                padding: 10px 15px;
                border: none;
                border-radius: 5px;
                cursor: pointer;
                margin: 5px;
                transition: background 0.3s;
            }
            button:hover { background: #1976d2; }
            pre {
                background: #f8f9fa;
                padding: 15px;
                border-radius: 5px;
                overflow-x: auto;
                max-height: 400px;
                overflow-y: auto;
                border: 1px solid #dee2e6;
            }
```

```
            .status-online { color: #4caf50; font-weight: bold; }
            .endpoint { margin: 5px 0; }
            .endpoint a { color: #2196f3; text-decoration: none; }
            .endpoint a:hover { text-decoration: underline; }
        </style>
    </head>
    <body>
        <div class="container">
            <h1>🚀 Serveur PeerJS Express</h1>

            <div class="stats">
                <h3>📊 Informations du Serveur</h3>
                <p><strong>Statut:</strong> <span class="status-online">✅ En
ligne</span></p>
                <p><strong>Port Express:</strong> ${EXPRESS_PORT}</p>
                <p><strong>Chemin PeerJS:</strong> ${PEERJS_PATH}</p>
                <p><strong>Clé PeerJS:</strong> webrtc-app-express</p>
                <p><strong>Démarré:</strong>
${stats.startTime.toLocaleString('fr-FR')}</p>
            </div>

            <h3>🔗 Endpoints Disponibles</h3>
            <div class="endpoint">📊 <a href="/api/stats">Statistiques
Détaillées</a></div>
            <div class="endpoint">💓 <a href="/api/health">Health
Check</a></div>
            <div class="endpoint">👥 <a href="/api/clients">Liste des
Clients</a></div>
            <div class="endpoint">🏠 <a href="/api/rooms">Gestion des
Salles</a></div>
            <div class="endpoint">✅ <a href="/api/test">Test de
Connectivité</a></div>

            <h3>🎮 Actions Rapides</h3>
            <button onclick="loadData('/api/health')">Health Check</button>
            <button onclick="loadData('/api/stats')">Statistiques</button>
            <button onclick="loadData('/api/clients')">Clients
Connectés</button>
            <button onclick="loadData('/api/test')">Test API</button>
            <button onclick="location.reload()">🔄 Actualiser</button>

            <div id="result"></div>

            <script>
                async function loadData(endpoint) {
                    try {
                        document.getElementById('result').innerHTML = '<p>⏳
Chargement...</p>';
                        const response = await fetch(endpoint);
```

```
                        const data = await response.json();
                        document.getElementById('result').innerHTML =
                            '<h4>📡 Réponse de ' + endpoint + ':</h4><pre>' +
                            JSON.stringify(data, null, 2) + '</pre>';
                    } catch (error) {
                        document.getElementById('result').innerHTML =
                            '<h4>❌ Erreur:</h4><pre>Erreur: ' + error.message
+ '</pre>';
                    }
                }

                // Auto-refresh des stats toutes les 30 secondes si affiché
                setInterval(() => {
                    const result = document.getElementById('result');
                    if (result.innerHTML.includes('/api/stats')) {
                        loadData('/api/stats');
                    }
                }, 30000);

                // Charger les stats au démarrage
                setTimeout(() => loadData('/api/stats'), 1000);
            </script>
        </div>
    </body>
    </html>
    `);
});

// Health check
app.get('/api/health', (req, res) => {
    res.json({
        status: 'healthy',
        timestamp: new Date().toISOString(),
        uptime: Date.now() - stats.startTime.getTime(),
        port: EXPRESS_PORT,
        peerjs_path: PEERJS_PATH
    });
});

// Statistiques complètes
app.get('/api/stats', (req, res) => {
    const uptime = Date.now() - stats.startTime.getTime();

    res.json({
        server: {
            ...stats,
            uptime: uptime,
            uptimeFormatted: formatUptime(uptime),
            port: EXPRESS_PORT,
```

```
            peerjs_path: PEERJS_PATH,
            memoryUsage: process.memoryUsage()
        },
        clients: {
            total: connectedClients.size,
            list: Array.from(connectedClients.entries()).map(([id, client]) =>
({
                id,
                connectedAt: client.connectedAt,
                lastActivity: client.lastActivity
            }))
        }
    });
});

// Liste des clients
app.get('/api/clients', (req, res) => {
    const clients = Array.from(connectedClients.entries()).map(([id, client])
=> ({
        id,
        connectedAt: client.connectedAt,
        lastActivity: client.lastActivity,
        duration: Date.now() - client.connectedAt.getTime()
    }));

    res.json({
        total: clients.length,
        clients: clients
    });
});

// Gestion des salles (basique)
app.get('/api/rooms', (req, res) => {
    res.json({
        total: 0,
        rooms: [],
        message: 'Fonctionnalité de salles disponible - pas de salles actives'
    });
});

// Test de connectivité
app.get('/api/test', (req, res) => {
    res.json({
        message: 'Serveur PeerJS Express fonctionnel !',
        timestamp: new Date().toISOString(),
        server_info: {
            port: EXPRESS_PORT,
            peerjs_path: PEERJS_PATH,
            node_version: process.version,
```

```
            uptime: formatUptime(Date.now() - stats.startTime.getTime())
        },
        endpoints: [
            'GET /',
            'GET /api/health',
            'GET /api/stats',
            'GET /api/clients',
            'GET /api/rooms',
            'GET /api/test'
        ]
    });
});

// Gestion d'erreur 404
app.use((req, res) => {
    res.status(404).json({
        error: 'Endpoint non trouvé',
        requested: req.path,
        available_endpoints: [
            '/',
            '/api/health',
            '/api/stats',
            '/api/clients',
            '/api/rooms',
            '/api/test'
        ]
    });
});

// ===== ÉVÉNEMENTS PEERJS =====

// Gestion des connexions PeerJS
peerServer.on('connection', (client) => {
    const clientInfo = {
        id: client.getId(),
        connectedAt: new Date(),
        lastActivity: new Date()
    };

    connectedClients.set(client.getId(), clientInfo);
    stats.totalConnections++;
    stats.currentConnections = connectedClients.size;

    console.log(`[EXPRESS-CONNECT] Client ${client.getId()} connecté (Total:
${connectedClients.size})`);

    // Événements du client
    client.on('disconnect', () => {
        connectedClients.delete(client.getId());
```

```
        stats.currentConnections = connectedClients.size;
        console.log(`[EXPRESS-DISCONNECT] Client ${client.getId()} déconnecté
(Total: ${connectedClients.size})`);
    });

    client.on('error', (error) => {
        console.error(`[EXPRESS-ERROR] Client ${client.getId()}:`,
error.message);
    });
});

peerServer.on('message', (client, message) => {
    stats.messagesHandled++;

    // Mettre à jour l'activité du client
    const clientInfo = connectedClients.get(client.getId());
    if (clientInfo) {
        clientInfo.lastActivity = new Date();
    }

    console.log(`[EXPRESS-MESSAGE] ${client.getId()} -> ${message.type}`);
});

// ===== FONCTIONS UTILITAIRES =====

function formatUptime(milliseconds) {
    const seconds = Math.floor(milliseconds / 1000);
    const minutes = Math.floor(seconds / 60);
    const hours = Math.floor(minutes / 60);
    const days = Math.floor(hours / 24);

    if (days > 0) return `${days}j ${hours % 24}h ${minutes % 60}m`;
    if (hours > 0) return `${hours}h ${minutes % 60}m ${seconds % 60}s`;
    if (minutes > 0) return `${minutes}m ${seconds % 60}s`;
    return `${seconds}s`;
}

// ===== DÉMARRAGE DU SERVEUR EXPRESS =====

const server = app.listen(EXPRESS_PORT, () => {
    console.log('\n🚀 Serveur PeerJS Express Démarré !');
    console.log(`📡 URL: http://localhost:${EXPRESS_PORT}/`);
    console.log(`🔗 PeerJS: http://localhost:${EXPRESS_PORT}${PEERJS_PATH}`);
    console.log(`📊 API: http://localhost:${EXPRESS_PORT}/api/`);
    console.log(`🌐 Dashboard: http://localhost:${EXPRESS_PORT}/`);
    console.log(`🔑 Clé: webrtc-app-express`);
    console.log('\n✋ Appuyez sur CTRL+C pour arrêter\n');
});
```

```
// Gestion propre de l'arrêt
process.on('SIGINT', () => {
    console.log('\n🛑 Arrêt du serveur Express...');
    server.close(() => {
        console.log('✅ Serveur Express fermé proprement');
        process.exit(0);
    });
});

process.on('SIGTERM', () => {
    console.log('🔄 Signal SIGTERM reçu, arrêt en cours...');
    server.close(() => {
        process.exit(0);
    });
});

module.exports = app;
```

3.2 Scripts de Gestion et de Test

- Package.json avec Scripts

nano package.json

```
root@sergio:/home/sergio/peerjs-basic-server# nano package.json
root@sergio:/home/sergio/peerjs-basic-server#
```

```
{
  "name": "peerjs-server-webrtc",
  "version": "1.0.0",
  "description": "Serveur PeerJS personnalisé pour application WebRTC",
  "main": "server-express.js",
  "scripts": {
    "start": "node server-express.js",
    "start:basic": "node server-basic.js",
    "dev": "nodemon server-express.js",
    "dev:basic": "nodemon server-basic.js",
    "test": "jest",
    "test:watch": "jest --watch",
    "logs": "tail -f logs/peerjs-*.log",
    "clean": "rm -rf logs/*.log",
    "install:global": "npm install -g peer",
    "stop": "pkill -f 'node.*server'"
  },
  "keywords": ["webrtc", "peerjs", "signaling", "express"],
  "author": "Votre Nom",
  "license": "MIT",
  "dependencies": {
```

```
    "peer": "^1.0.0",
    "express": "^4.18.2",
    "cors": "^2.8.5",
    "helmet": "^7.0.0",
    "morgan": "^1.10.0",
    "express-rate-limit": "^6.7.1",
    "ws": "^8.13.0",
    "uuid": "^9.0.0"
  },
  "devDependencies": {
    "nodemon": "^3.0.1",
    "jest": "^29.5.0",
    "supertest": "^6.3.3"
  }
}
```

- Script de Démarrage Automatique (scripts/start-servers.sh)

nano scripts/start-servers.sh

```
root@sergio:/home/sergio/peerjs-basic-server# nano scripts/start-servers.sh
root@sergio:/home/sergio/peerjs-basic-server#
```

```
#!/bin/bash

echo "=== Démarrage Corrigé des Serveurs PeerJS ==="

# Aller dans le répertoire du projet
cd /home/sergio/peerjs-basic-server

# Fonction de nettoyage
cleanup() {
    echo "🛑 Arrêt des serveurs..."
    pkill -f "node.*server" 2>/dev/null || true
    exit 0
}

# Capturer CTRL+C
trap cleanup SIGINT SIGTERM

# Tuer les anciens processus
echo "🧹 Nettoyage des anciens processus..."
pkill -f "node.*server" 2>/dev/null || true
sleep 3

# Vérifier que les ports sont libres
echo "🔍 Vérification des ports..."
if netstat -tln | grep -q ":9000 "; then
    echo "⚠️  Port 9000 occupé, tentative de libération..."
```

```
    sudo fuser -k 9000/tcp 2>/dev/null || true
    sleep 2
fi

if netstat -tln | grep -q ":9002 "; then
    echo "⚠️ Port 9002 occupé, tentative de libération..."
    sudo fuser -k 9002/tcp 2>/dev/null || true
    sleep 2
fi

# Menu de choix
echo "Choisissez le serveur à démarrer:"
echo "1) 📡 Serveur Basique (Sans Express) - Port 9000"
echo "2) 🚀 Serveur Express (Avec API) - Port 9000"
echo "3) 🔄 Les deux serveurs (Ports 9000 + 9002)"
echo "4) ❌ Annuler"

read -p "Votre choix (1-4): " choice

case $choice in
    1)
        echo "🚀 Démarrage du serveur basique sur le port 9000..."
        node server-basic.js
        ;;
    2)
        echo "🚀 Démarrage du serveur Express sur le port 9002..."
        node server-express.js
        ;;
    3)
        echo "🚀 Démarrage des deux serveurs..."

        # Démarrer le serveur basique en arrière-plan
        echo "📡 Démarrage du serveur basique (port 9000)..."
        node server-basic.js &
        BASIC_PID=$!

        # Attendre que le serveur basique démarre
        sleep 3

        # Vérifier que le serveur basique est bien démarré
        if ! kill -0 $BASIC_PID 2>/dev/null; then
            echo "❌ Échec du démarrage du serveur basique"
            exit 1
        fi

        echo "✅ Serveur basique démarré (PID: $BASIC_PID)"

        # Démarrer le serveur Express en arrière-plan
        echo "🚀 Démarrage du serveur Express (port 9002)..."
```

```
        node server-express.js &
        EXPRESS_PID=$!

        # Attendre que le serveur Express démarre
        sleep 3

        # Vérifier que le serveur Express est bien démarré
        if ! kill -0 $EXPRESS_PID 2>/dev/null; then
            echo "❌ Échec du démarrage du serveur Express"
            kill $BASIC_PID 2>/dev/null || true
            exit 1
        fi

        echo "✅ Serveur Express démarré (PID: $EXPRESS_PID)"
        echo ""
        echo "🎉 Les deux serveurs sont opérationnels !"
        echo "🔗 Basique: http://localhost:9000/peerjs"
        echo "🔗 Express: http://localhost:9002/"
        echo "📊 Dashboard: http://localhost:9002/"
        echo ""
        echo "🛑 Appuyez sur CTRL+C pour arrêter les deux serveurs"

        # Attendre les processus
        wait $BASIC_PID $EXPRESS_PID
        ;;
    4)
        echo "❌ Opération annulée"
        exit 0
        ;;
    *)
        echo "❌ Choix invalide"
        exit 1
        ;;
esac
```

4. Tests et Validation

- Commandes de Déploiement

chmod +x scripts/*.sh

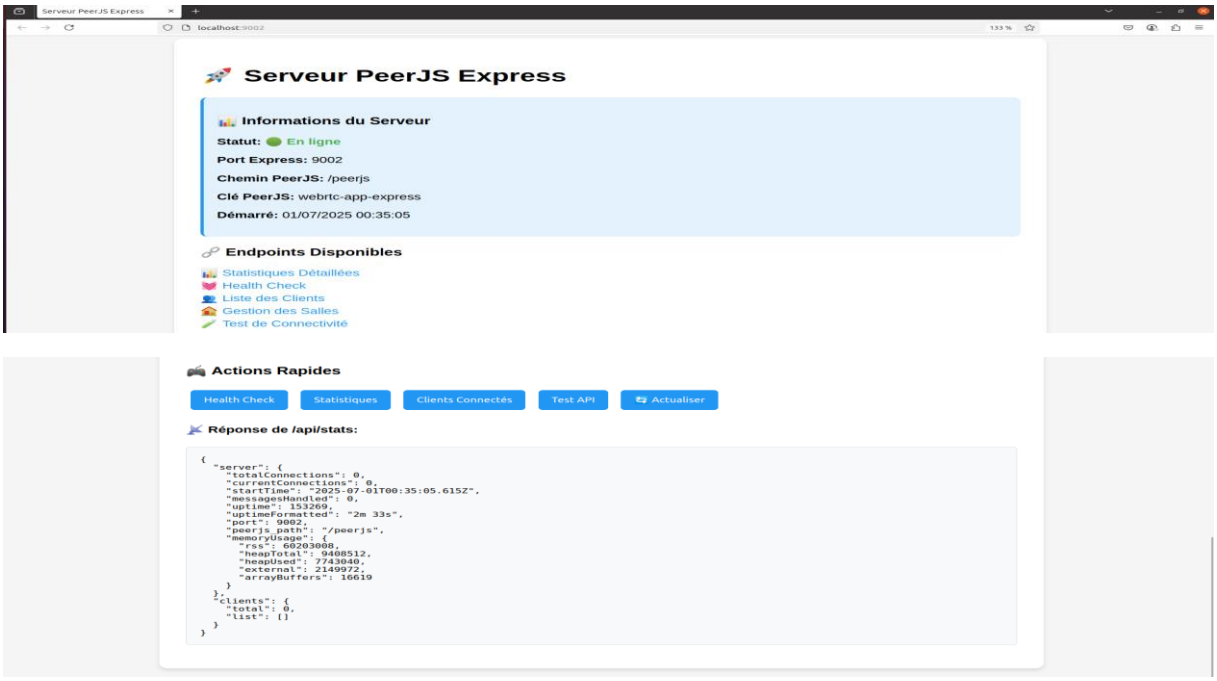
```
root@sergio:/home/sergio/peerjs-basic-server# chmod +x scripts/*.sh
root@sergio:/home/sergio/peerjs-basic-server#
```

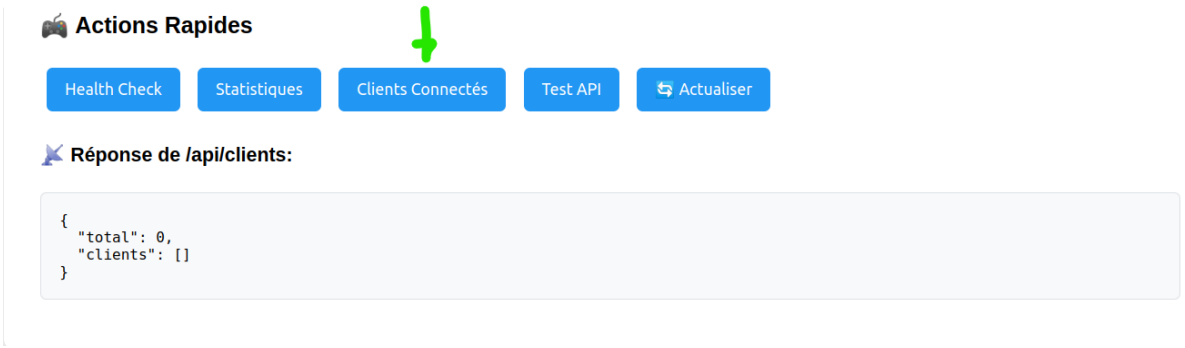
5. Démarrer les serveurs

./scripts/start-servers.sh

On test le serveur Expresse (avec API)

Sur le navigateur on accede à : http://localhost:9002





- On demarre les deux serveurs

```
root@sergio:/home/sergio/peerjs-basic-server# ./scripts/start-servers.sh
=== Démarrage Corrigé des Serveurs PeerJS ===
Nettoyage des anciens processus...
Vérification des ports...
Choisissez le serveur à démarrer:
1) Serveur Basique (Sans Express) - Port 9000
2) Serveur Express (Avec API) - Port 9000
3) Les deux serveurs (Ports 9000 + 9002)
4) Annuler
Votre choix (1-4): 3
Démarrage des deux serveurs...
Démarrage du serveur basique (port 9000)...
[2025-07-01T00:40:00.261Z] [SERVER_START] Serveur PeerJS démarré
Data: { port: 9000, path: '/peerjs', key: 'webrtc-app', pid: 17521 }

Serveur PeerJS Basique Démarré !
URL: http://localhost:9000/peerjs
Clé: webrtc-app
Logs: /home/sergio/peerjs-basic-server/logs

Appuyez sur CTRL+C pour arrêter

[2025-07-01T00:40:00.269Z] [STATS_SERVER] Serveur de statistiques démarré sur le port 9001
Stats API: http://localhost:9001/stats
Serveur basique démarré (PID: 17521)
Démarrage du serveur Express (port 9002)...
Démarrage du serveur Express...

Serveur PeerJS Express Démarré !
URL: http://localhost:9002/
PeerJS: http://localhost:9002/peerjs
API: http://localhost:9002/api/
Dashboard: http://localhost:9002/
Clé: webrtc-app-express

Appuyez sur CTRL+C pour arrêter
```

- Arrêt de tous les serveurs

pkill -f "node.*server" 2>/dev/null || true

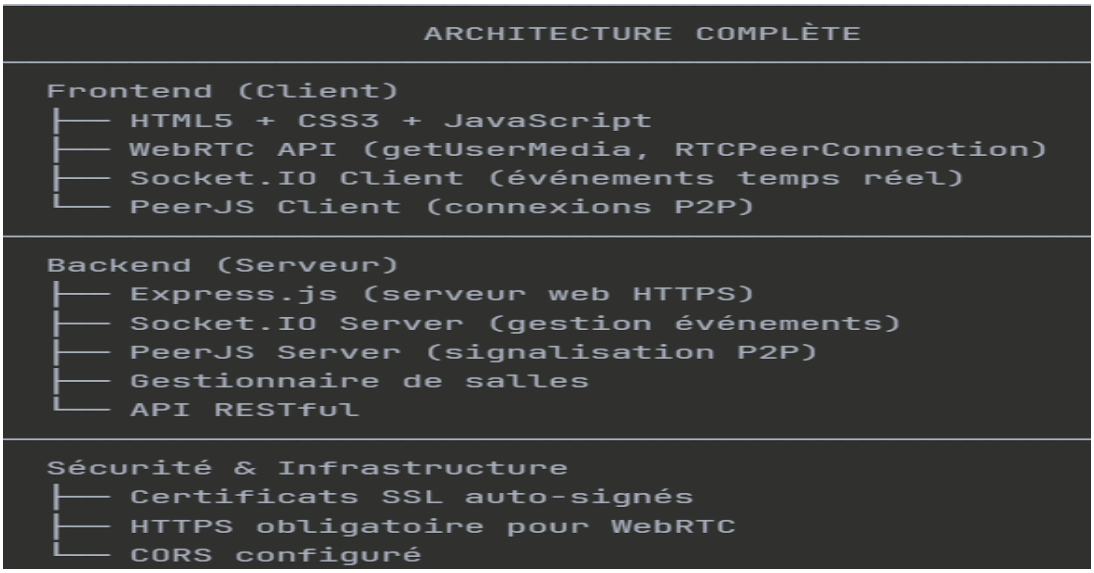
sleep 3

```
root@sergio:/home/sergio/peerjs-basic-server# pkill -f "node.*server" 2>/dev/null || true
root@sergio:/home/sergio/peerjs-basic-server# sleep 3
root@sergio:/home/sergio/peerjs-basic-server#
```

3. Application Complète WebRTC - PeerJS + Socket.IO

3.1. Architecture de l'Application

Vue d'ensemble de l'architecture.



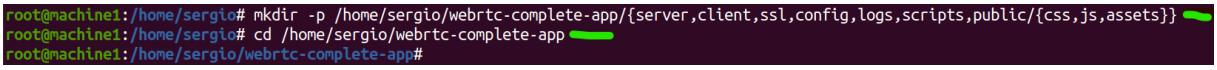
3.2 Création du Projet Complet

3.2.1 Structure du projet

Créons la structure complète du projet par la commande suivante

mkdir -p /home/sergio/webrtc-complete-app/{server,client,ssl,config,logs,scripts,public/{css,js,assets}}

cd /home/sergio/webrtc-complete-app



- **Structure finale**

webrtc-complete-app/

├── server/

│ ├── app.js # Serveur principal

│ ├── socketHandler.js # Gestion Socket.IO

│ ├── peerHandler.js # Gestion PeerJS

│ └── roomManager.js # Gestionnaire de salles

├── client/

│ └── webrtc-client.js # Client WebRTC

```
├── public/
│   ├── index.html          # Interface principale
│   ├── css/style.css       # Styles
│   └── js/app.js           # Application frontend
├── ssl/                    # Certificats SSL
├── config/                 # Configuration
├── logs/                   # Logs
└── scripts/                # Scripts utilitaires
```

3.2.2 Initialisation du projet

Initialisant le package.json par la commande suivante :

npm init -y

```
root@machine1:/home/sergio/webrtc-complete-app# npm init -y
Wrote to /home/sergio/webrtc-complete-app/package.json:

{
  "name": "webrtc-complete-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

```
root@machine1:/home/sergio/webrtc-complete-app# cat > package.json << 'EOF'
> {
>   "name": "webrtc-complete-app",
>   "version": "1.0.0",
>   "description": "Application WebRTC complète avec PeerJS et Socket.IO",
>   "main": "server/app.js",
>   "scripts": {
>     "start": "node server/app.js",
>     "dev": "nodemon server/app.js",
>     "test": "node scripts/test-app.js",
>     "generate-ssl": "node scripts/generate-ssl.js",
>     "clean": "rm -rf logs/*.log ssl/*.pem"
>   },
>   "keywords": ["webrtc", "peerjs", "socket.io", "video-chat"],
>   "author": "Sergio",
>   "license": "MIT",
>   "dependencies": {
>     "express": "^4.18.2",
>     "socket.io": "^4.7.2",
>     "peer": "^1.0.0",
>     "https": "^1.0.0",
>     "fs": "^0.0.1-security",
>     "path": "^0.12.7",
>     "cors": "^2.8.5",
>     "helmet": "^7.0.0",
>     "uuid": "^9.0.0"
>   },
>   "devDependencies": {
>     "nodemon": "^3.0.1"
>   }
> }
> EOF
```

Voici la commande qui introduit le nouveau contenu du fichier package.son que vous allez exécuter comme ci-dessous.

```
cat > package.json << 'EOF'
{
  "name": "webrtc-complete-app",
  "version": "1.0.0",
  "description": "Application WebRTC complète avec PeerJS et Socket.IO",
  "main": "server/app.js",
```

```
  "scripts": {
    "start": "node server/app.js",
    "dev": "nodemon server/app.js",
    "test": "node scripts/test-app.js",
    "generate-ssl": "node scripts/generate-ssl.js",
    "clean": "rm -rf logs/*.log ssl/*.pem"
  },
  "keywords": ["webrtc", "peerjs", "socket.io", "video-chat"],
  "author": "Sergio",
  "license": "MIT",
  "dependencies": {
    "express": "^4.18.2",
    "socket.io": "^4.7.2",
    "peer": "^1.0.0",
    "https": "^1.0.0",
    "fs": "^0.0.1-security",
    "path": "^0.12.7",
    "cors": "^2.8.5",
    "helmet": "^7.0.0",
    "uuid": "^9.0.0"
  },
  "devDependencies": {
    "nodemon": "^3.0.1"
  }
}
EOF
```

Installons ensuite les dépendances

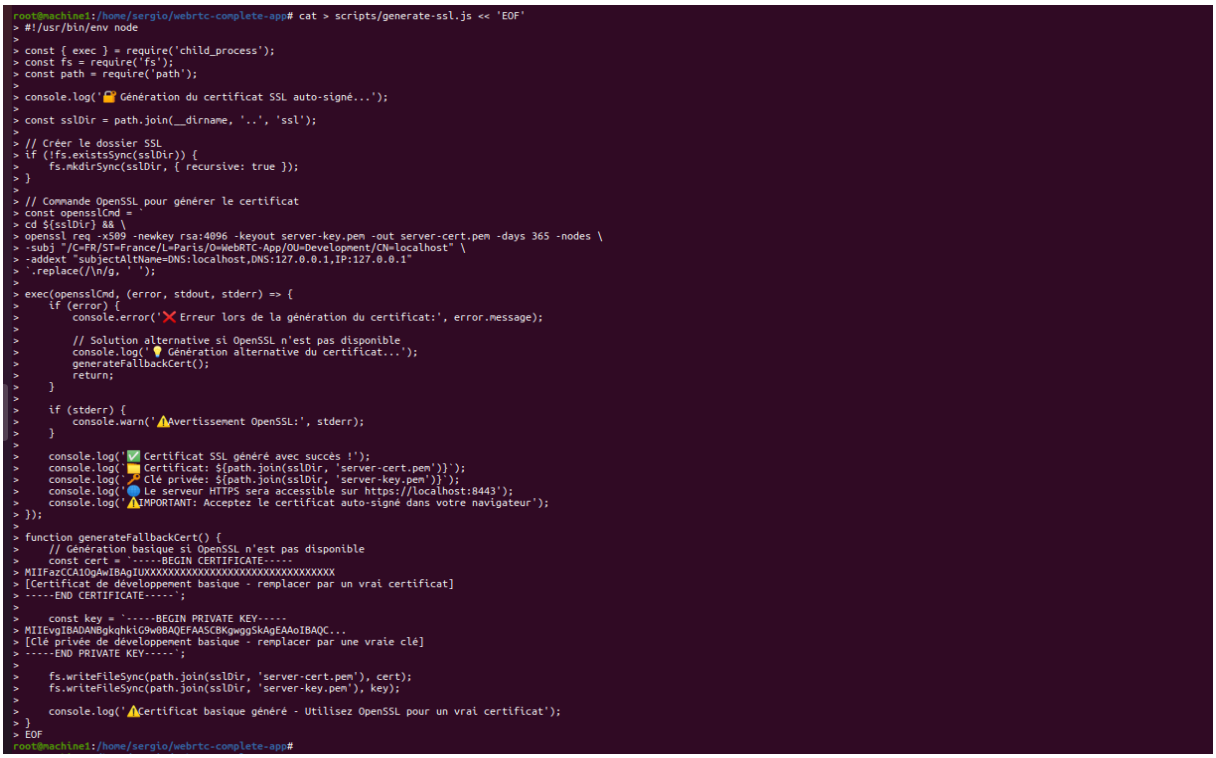
npm install

```
root@machine1:/home/sergio/webrtc-complete-app# npm install
npm warn deprecated node-domexception@1.0.0: Use your platform's native DOMException instead

added 164 packages, and audited 165 packages in 44s

25 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@machine1:/home/sergio/webrtc-complete-app#
```

3.3 Génération du Certificat SSL Auto-signé

3.3.1 Script de génération SSL

Créons le **scripts/generate-ssl.js**



Voici la commande d'automatisation de la création de **scripts/generate-ssl.js**

```
cat > scripts/generate-ssl.js << 'EOF'
#!/usr/bin/env node

const { exec } = require('child_process');
const fs = require('fs');
const path = require('path');

console.log('🔐 Génération du certificat SSL auto-signé...');

const sslDir = path.join(__dirname, '..', 'ssl');

// Créer le dossier SSL
if (!fs.existsSync(sslDir)) {
    fs.mkdirSync(sslDir, { recursive: true });
}

// Commande OpenSSL pour générer le certificat
const opensslCmd = `
cd ${sslDir} && \
openssl req -x509 -newkey rsa:4096 -keyout server-key.pem -out server-cert.pem
-days 365 -nodes \
```

```
-subj "/C=SN/ST=Senegal/L=Dakar/O=WebRTC-App/OU=Development/CN=localhost" \
-addext "subjectAltName=DNS:localhost,DNS:127.0.0.1,IP:127.0.0.1"
`.replace(/\n/g, ' ');

exec(opensslCmd, (error, stdout, stderr) => {
    if (error) {
        console.error('❌ Erreur lors de la génération du certificat:',
error.message);

        // Solution alternative si OpenSSL n'est pas disponible
        console.log('💡 Génération alternative du certificat...');
        generateFallbackCert();
        return;
    }

    if (stderr) {
        console.warn('⚠️ Avertissement OpenSSL:', stderr);
    }

    console.log('✅ Certificat SSL généré avec succès !');
    console.log(`📄 Certificat: ${path.join(sslDir, 'server-cert.pem')}`);
    console.log(`🔑 Clé privée: ${path.join(sslDir, 'server-key.pem')}`);
    console.log('🌐 Le serveur HTTPS sera accessible sur
https://localhost:8443');
    console.log('⚠️ IMPORTANT: Acceptez le certificat auto-signé dans votre
navigateur');
});

function generateFallbackCert() {
    // Génération basique si OpenSSL n'est pas disponible
    const cert = `-----BEGIN CERTIFICATE-----
MIIFazCCA1OgAwIBAgIUXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
[Certificat de développement basique - remplacer par un vrai certificat]
-----END CERTIFICATE-----`;

    const key = `-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC...
[Clé privée de développement basique - remplacer par une vraie clé]
-----END PRIVATE KEY-----`;

    fs.writeFileSync(path.join(sslDir, 'server-cert.pem'), cert);
    fs.writeFileSync(path.join(sslDir, 'server-key.pem'), key);

    console.log('⚠️ Certificat basique généré - Utilisez OpenSSL pour un vrai
certificat');
}
EOF
```

chmod +x scripts/generate-ssl.js

```
root@machine1:/home/sergio/webrtc-complete-app# chmod +x scripts/generate-ssl.js
root@machine1:/home/sergio/webrtc-complete-app#
```

3.3.2 Génération du certificat

On génère le certificat SSL par la commande suivante

node scripts/generate-ssl.js

```
root@machine1:/home/sergio/webrtc-complete-app# node scripts/generate-ssl.js
🔐 Génération du certificat SSL auto-signé...
⚠️Avertissement OpenSSL: Generating a RSA private key
.....................................++++
.............................++++
writing new private key to 'server-key.pem'
-----

✅ Certificat SSL généré avec succès !
📜 Certificat: /home/sergio/webrtc-complete-app/ssl/server-cert.pem
🔑 Clé privée: /home/sergio/webrtc-complete-app/ssl/server-key.pem
🌐 Le serveur HTTPS sera accessible sur https://localhost:8443
⚠️IMPORTANT: Acceptez le certificat auto-signé dans votre navigateur
root@machine1:/home/sergio/webrtc-complete-app#
```

Ou manuellement avec OpenSSL

mkdir -p ssl

cd ssl openssl req -x509 -newkey rsa:2048 -keyout server-key.pem -out server-cert.pem -days 365 -nodes \ -subj "/C=SN/ST=Senegal/L=Dakar/O=WebRTC-App/CN=localhost"

cd ..

3.4 Configuration du Serveur

Configuration principale (**config/app-config.js**)

nano config/app-config.js

```
root@machine1:/home/sergio/webrtc-complete-app# nano config/app-config.js
root@machine1:/home/sergio/webrtc-complete-app#
```

```
module.exports = {
    // Configuration du serveur
    server: {
        port: process.env.PORT || 8443,
        host: process.env.HOST || 'localhost',
        ssl: {
            enabled: true,
            certPath: './ssl/server-cert.pem',
            keyPath: './ssl/server-key.pem'
        }
    },

    // Configuration Socket.IO
    socketio: {
        cors: {
```

```
            origin: ["https://localhost:8443", "https://127.0.0.1:8443"],
            methods: ["GET", "POST"],
            credentials: true
        },
        pingTimeout: 60000,
        pingInterval: 25000
    },

    // Configuration PeerJS
    peerjs: {
        path: '/peerjs',
        key: 'webrtc-complete-app',
        allow_discovery: true,
        debug: 1
    },

    // Configuration des salles
    rooms: {
        maxClients: 10,
        defaultTTL: 3600000, // 1 heure
        cleanupInterval: 300000 // 5 minutes
    },

    // Configuration de sécurité
    security: {
        helmet: {
            contentSecurityPolicy: {
                directives: {
                    defaultSrc: ["'self'"],
                    scriptSrc: ["'self'", "'unsafe-inline'",
"https://unpkg.com"],
                    styleSrc: ["'self'", "'unsafe-inline'"],
                    connectSrc: ["'self'", "wss:", "ws:"],
                    mediaSrc: ["'self'"]
                }
            }
        }
    }
};
```

- Gestionnaire de salles (**server/roomManager.js**)

nano server/roomManager.js

```
root@machine1:/home/sergio/webrtc-complete-app# nano server/roomManager.js
root@machine1:/home/sergio/webrtc-complete-app#
```

```
const { v4: uuidv4 } = require('uuid');

class RoomManager {
    constructor() {
```

```
        this.rooms = new Map();
        this.clients = new Map(); // clientId -> roomId
        this.stats = {
            totalRooms: 0,
            totalClients: 0,
            peakConcurrentRooms: 0
        };

        // Nettoyage automatique des salles vides
        setInterval(() => this.cleanupEmptyRooms(), 5 * 60 * 1000); // 5
minutes
    }

    createRoom(roomId = null, maxClients = 10, metadata = {}) {
        const id = roomId || uuidv4();

        if (this.rooms.has(id)) {
            throw new Error(`Salle ${id} existe déjà`);
        }

        const room = {
            id,
            createdAt: new Date(),
            lastActivity: new Date(),
            maxClients,
            clients: new Map(), // clientId -> clientInfo
            metadata: {
                name: metadata.name || `Salle ${id.substring(0, 8)}`,
                description: metadata.description || '',
                isPrivate: metadata.isPrivate || false,
                password: metadata.password || null,
                ...metadata
            },
            stats: {
                totalConnections: 0,
                messagesExchanged: 0,
                callsStarted: 0
            }
        };

        this.rooms.set(id, room);
        this.stats.totalRooms++;
        this.stats.peakConcurrentRooms =
Math.max(this.stats.peakConcurrentRooms, this.rooms.size);

        console.log(`[ROOM] Salle créée: ${id} (${room.metadata.name})`);
        return room;
    }
```

```
    joinRoom(roomId, clientId, clientInfo = {}) {
        const room = this.rooms.get(roomId);
        if (!room) {
            throw new Error(`Salle ${roomId} n'existe pas`);
        }

        if (room.clients.size >= room.maxClients) {
            throw new Error(`Salle ${roomId} pleine (${room.maxClients}
clients max)`);
        }

        if (room.clients.has(clientId)) {
            throw new Error(`Client ${clientId} déjà dans la salle
${roomId}`);
        }

        // Ajouter le client à la salle
        const client = {
            id: clientId,
            joinedAt: new Date(),
            lastActivity: new Date(),
            info: {
                userAgent: clientInfo.userAgent || 'Unknown',
                peerId: clientInfo.peerId || null,
                nickname: clientInfo.nickname || `User_${clientId.substring(0,
6)}`,
                ...clientInfo
            }
        };

        room.clients.set(clientId, client);
        this.clients.set(clientId, roomId);
        room.lastActivity = new Date();
        room.stats.totalConnections++;
        this.stats.totalClients++;

        console.log(`[ROOM] Client ${clientId} rejoint la salle ${roomId}
(${room.clients.size}/${room.maxClients})`);

        return {
            room: this.getRoomInfo(roomId),
            client: client,
            otherClients: Array.from(room.clients.values()).filter(c => c.id
!== clientId)
        };
    }

    leaveRoom(clientId) {
        const roomId = this.clients.get(clientId);
```

```
        if (!roomId) {
            return null; // Client pas dans une salle
        }

        const room = this.rooms.get(roomId);
        if (!room) {
            this.clients.delete(clientId);
            return null;
        }

        const client = room.clients.get(clientId);
        room.clients.delete(clientId);
        this.clients.delete(clientId);
        room.lastActivity = new Date();

        console.log(`[ROOM] Client ${clientId} quitte la salle ${roomId}
(${room.clients.size}/${room.maxClients})`);

        // Supprimer la salle si vide
        if (room.clients.size === 0) {
            this.rooms.delete(roomId);
            console.log(`[ROOM] Salle ${roomId} supprimée (vide)`);
        }

        return {
            roomId,
            client,
            remainingClients: room.clients.size
        };
    }

    getRoomInfo(roomId) {
        const room = this.rooms.get(roomId);
        if (!room) return null;

        return {
            id: room.id,
            createdAt: room.createdAt,
            lastActivity: room.lastActivity,
            clientCount: room.clients.size,
            maxClients: room.maxClients,
            metadata: room.metadata,
            stats: room.stats,
            clients: Array.from(room.clients.values()).map(client => ({
                id: client.id,
                joinedAt: client.joinedAt,
                nickname: client.info.nickname,
                peerId: client.info.peerId
            }))
```

```
        };
    }

    getAllRooms() {
        return Array.from(this.rooms.keys()).map(roomId =>
this.getRoomInfo(roomId));
    }

    getClientRoom(clientId) {
        const roomId = this.clients.get(clientId);
        return roomId ? this.getRoomInfo(roomId) : null;
    }

    updateClientActivity(clientId) {
        const roomId = this.clients.get(clientId);
        if (!roomId) return;

        const room = this.rooms.get(roomId);
        if (!room) return;

        const client = room.clients.get(clientId);
        if (client) {
            client.lastActivity = new Date();
            room.lastActivity = new Date();
        }
    }

    incrementRoomStat(roomId, statName) {
        const room = this.rooms.get(roomId);
        if (room && room.stats[statName] !== undefined) {
            room.stats[statName]++;
        }
    }

    cleanupEmptyRooms() {
        const cutoffTime = new Date(Date.now() - 60 * 60 * 1000); // 1 heure

        for (const [roomId, room] of this.rooms.entries()) {
            if (room.clients.size === 0 && room.lastActivity < cutoffTime) {
                this.rooms.delete(roomId);
                console.log(`[CLEANUP] Salle ${roomId} supprimée (inactive)`);
            }
        }
    }

    getStats() {
        return {
            ...this.stats,
            currentRooms: this.rooms.size,
```

```
            currentClients: this.clients.size,
            rooms: this.getAllRooms()
        };
    }
}

module.exports = RoomManager;
```

- Gestionnaire Socket.IO (**server/socketHandler.js**)

nano server/socketHandler.js

```
root@machine1:/home/sergio/webrtc-complete-app# nano server/socketHandler.js
root@machine1:/home/sergio/webrtc-complete-app#
```

```
const RoomManager = require('./roomManager');

class SocketHandler {
    constructor(io, roomManager) {
        this.io = io;
        this.roomManager = roomManager;
        this.setupEventHandlers();
    }

    setupEventHandlers() {
        this.io.on('connection', (socket) => {
            console.log(`[SOCKET] Client connecté: ${socket.id}`);

            // Événements de gestion des salles
            socket.on('create-room', (data, callback) =>
this.handleCreateRoom(socket, data, callback));
            socket.on('join-room', (data, callback) =>
this.handleJoinRoom(socket, data, callback));
            socket.on('leave-room', (callback) => this.handleLeaveRoom(socket,
callback));
            socket.on('get-room-info', (callback) =>
this.handleGetRoomInfo(socket, callback));
            socket.on('list-rooms', (callback) => this.handleListRooms(socket,
callback));

            // Événements WebRTC
            socket.on('webrtc-offer', (data) => this.handleWebRTCOffer(socket,
data));
            socket.on('webrtc-answer', (data) =>
this.handleWebRTCAnswer(socket, data));
            socket.on('webrtc-ice-candidate', (data) =>
this.handleICECandidate(socket, data));
            socket.on('start-call', (data) => this.handleStartCall(socket,
data));
```

```
            socket.on('end-call', (data) => this.handleEndCall(socket, data));

            // Événements de chat
            socket.on('chat-message', (data) => this.handleChatMessage(socket,
data));
            socket.on('typing-start', (data) => this.handleTypingStart(socket,
data));
            socket.on('typing-stop', (data) => this.handleTypingStop(socket,
data));

            // Événements de partage d'écran
            socket.on('screen-share-start', (data) =>
this.handleScreenShareStart(socket, data));
            socket.on('screen-share-stop', (data) =>
this.handleScreenShareStop(socket, data));

            // Événements de présence
            socket.on('update-status', (data) =>
this.handleUpdateStatus(socket, data));

            // Déconnexion
            socket.on('disconnect', () => this.handleDisconnect(socket));
        });
    }

    handleCreateRoom(socket, data, callback) {
        try {
            const { roomId, maxClients = 10, metadata = {} } = data;
            const room = this.roomManager.createRoom(roomId, maxClients,
metadata);

            callback({
                success: true,
                room: this.roomManager.getRoomInfo(room.id)
            });

            // Notifier tous les clients de la nouvelle salle
            this.io.emit('room-created',
this.roomManager.getRoomInfo(room.id));

        } catch (error) {
            callback({
                success: false,
                error: error.message
            });
        }
    }

    handleJoinRoom(socket, data, callback) {
```

```
        try {
            const { roomId, clientInfo = {} } = data;
            const result = this.roomManager.joinRoom(roomId, socket.id,
clientInfo);

            // Rejoindre la room Socket.IO
            socket.join(roomId);

            callback({
                success: true,
                ...result
            });

            // Notifier les autres clients de la salle
            socket.to(roomId).emit('client-joined', {
                client: result.client,
                room: result.room
            });

            // Notifier tous les clients de la mise à jour de la salle
            this.io.emit('room-updated', result.room);

        } catch (error) {
            callback({
                success: false,
                error: error.message
            });
        }
    }

    handleLeaveRoom(socket, callback) {
        const result = this.roomManager.leaveRoom(socket.id);

        if (result) {
            socket.leave(result.roomId);

            // Notifier les autres clients
            socket.to(result.roomId).emit('client-left', {
                clientId: socket.id,
                remainingClients: result.remainingClients
            });

            // Notifier de la mise à jour de la salle si elle existe encore
            if (result.remainingClients > 0) {
                const roomInfo = this.roomManager.getRoomInfo(result.roomId);
                this.io.emit('room-updated', roomInfo);
            } else {
                this.io.emit('room-deleted', { roomId: result.roomId });
            }
```

```
        }

        if (callback) {
            callback({
                success: true,
                result: result
            });
        }
    }

    handleGetRoomInfo(socket, callback) {
        const room = this.roomManager.getClientRoom(socket.id);
        callback({
            success: true,
            room: room
        });
    }

    handleListRooms(socket, callback) {
        const rooms = this.roomManager.getAllRooms();
        callback({
            success: true,
            rooms: rooms
        });
    }

    handleWebRTCOffer(socket, data) {
        const { targetClientId, offer } = data;
        const room = this.roomManager.getClientRoom(socket.id);

        if (room) {
            socket.to(targetClientId).emit('webrtc-offer', {
                from: socket.id,
                offer: offer,
                roomId: room.id
            });

            this.roomManager.updateClientActivity(socket.id);
        }
    }

    handleWebRTCAnswer(socket, data) {
        const { targetClientId, answer } = data;
        const room = this.roomManager.getClientRoom(socket.id);

        if (room) {
            socket.to(targetClientId).emit('webrtc-answer', {
                from: socket.id,
                answer: answer,
```

```
                roomId: room.id
            });

            this.roomManager.updateClientActivity(socket.id);
        }
    }

    handleICECandidate(socket, data) {
        const { targetClientId, candidate } = data;
        const room = this.roomManager.getClientRoom(socket.id);

        if (room) {
            socket.to(targetClientId).emit('webrtc-ice-candidate', {
                from: socket.id,
                candidate: candidate,
                roomId: room.id
            });
        }
    }

    handleStartCall(socket, data) {
        const { targetClientId } = data;
        const room = this.roomManager.getClientRoom(socket.id);

        if (room) {
            socket.to(targetClientId).emit('call-incoming', {
                from: socket.id,
                roomId: room.id
            });

            this.roomManager.incrementRoomStat(room.id, 'callsStarted');
        }
    }

    handleEndCall(socket, data) {
        const { targetClientId } = data;
        const room = this.roomManager.getClientRoom(socket.id);

        if (room) {
            socket.to(targetClientId).emit('call-ended', {
                from: socket.id,
                roomId: room.id
            });
        }
    }

    handleChatMessage(socket, data) {
        const { message, timestamp } = data;
        const room = this.roomManager.getClientRoom(socket.id);
```

```
        if (room) {
            const chatData = {
                from: socket.id,
                message: message,
                timestamp: timestamp || new Date().toISOString(),
                roomId: room.id
            };

            // Envoyer à tous les clients de la salle
            this.io.to(room.id).emit('chat-message', chatData);

            this.roomManager.incrementRoomStat(room.id, 'messagesExchanged');
            this.roomManager.updateClientActivity(socket.id);
        }
    }

    handleTypingStart(socket, data) {
        const room = this.roomManager.getClientRoom(socket.id);
        if (room) {
            socket.to(room.id).emit('typing-start', {
                from: socket.id,
                roomId: room.id
            });
        }
    }

    handleTypingStop(socket, data) {
        const room = this.roomManager.getClientRoom(socket.id);
        if (room) {
            socket.to(room.id).emit('typing-stop', {
                from: socket.id,
                roomId: room.id
            });
        }
    }

    handleScreenShareStart(socket, data) {
        const room = this.roomManager.getClientRoom(socket.id);
        if (room) {
            socket.to(room.id).emit('screen-share-start', {
                from: socket.id,
                roomId: room.id
            });
        }
    }

    handleScreenShareStop(socket, data) {
        const room = this.roomManager.getClientRoom(socket.id);
```

```
        if (room) {
            socket.to(room.id).emit('screen-share-stop', {
                from: socket.id,
                roomId: room.id
            });
        }
    }

    handleUpdateStatus(socket, data) {
        const { status } = data;
        const room = this.roomManager.getClientRoom(socket.id);

        if (room) {
            socket.to(room.id).emit('status-updated', {
                from: socket.id,
                status: status,
                roomId: room.id
            });
        }
    }

    handleDisconnect(socket) {
        console.log(`[SOCKET] Client déconnecté: ${socket.id}`);
        this.handleLeaveRoom(socket);
    }
}

module.exports = SocketHandler;
```

- Serveur principal (**server/app.js**)

nano server/app.js

```
root@machine1:/home/sergio/webrtc-complete-app# nano server/app.js
root@machine1:/home/sergio/webrtc-complete-app#
```

```
#!/usr/bin/env node

const express = require('express');
const https = require('https');
const { Server } = require('socket.io');
const { PeerServer } = require('peer');
const fs = require('fs');
const path = require('path');
const cors = require('cors');
const helmet = require('helmet');

const config = require('../config/app-config');
const RoomManager = require('./roomManager');
```

```
const SocketHandler = require('./socketHandler');

console.log('🚀 Démarrage de l\'application WebRTC complète...');

// Créer l'application Express
const app = express();

// Configuration de sécurité avec Helmet CORRIGÉE
app.use(helmet({
    contentSecurityPolicy: {
        directives: {
            defaultSrc: ["'self'"],
            scriptSrc: ["'self'", "'unsafe-inline'", "https://unpkg.com"],
            scriptSrcAttr: ["'unsafe-inline'"], // AJOUT pour permettre
onclick
            styleSrc: ["'self'", "'unsafe-inline'"],
            connectSrc: ["'self'", "wss:", "ws:", "https:"],
            mediaSrc: ["'self'"],
            fontSrc: ["'self'"],
            imgSrc: ["'self'", "data:"],
            objectSrc: ["'none'"],
            frameAncestors: ["'none'"]
        }
    },
    crossOriginEmbedderPolicy: false // Nécessaire pour WebRTC
}));

// Configuration CORS
app.use(cors({
    origin: ["https://localhost:8443", "https://127.0.0.1:8443"],
    credentials: true
}));

// Middlewares
app.use(express.json({ limit: '10mb' }));
app.use(express.urlencoded({ extended: true }));

// IMPORTANT: Servir les fichiers statiques avec les bons types MIME
app.use(express.static('public', {
    setHeaders: (res, path) => {
        if (path.endsWith('.css')) {
            res.setHeader('Content-Type', 'text/css');
        } else if (path.endsWith('.js')) {
            res.setHeader('Content-Type', 'application/javascript');
        }
    }
}));

// Initialisation des gestionnaires
```

```
const roomManager = new RoomManager();

// Configuration SSL
let sslOptions;
try {
    sslOptions = {
        key: fs.readFileSync(config.server.ssl.keyPath),
        cert: fs.readFileSync(config.server.ssl.certPath)
    };
    console.log('✅ Certificats SSL chargés');
} catch (error) {
    console.error('❌ Erreur de chargement des certificats SSL:',
error.message);
    console.log('💡 Générez les certificats avec: node scripts/generate-
ssl.js');
    process.exit(1);
}

// Création du serveur HTTPS
const server = https.createServer(sslOptions, app);

// Configuration Socket.IO
const io = new Server(server, config.socketio);

// Intégration PeerJS
const peerServer = PeerServer(config.peerjs);
app.use(config.peerjs.path, peerServer);

// Initialisation du gestionnaire Socket.IO
const socketHandler = new SocketHandler(io, roomManager);

// ===== ROUTES API =====

// Page d'accueil
app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, '..', 'public', 'index.html'));
});

// API de santé
app.get('/api/health', (req, res) => {
    res.json({
        status: 'healthy',
        timestamp: new Date().toISOString(),
        ssl: true,
        services: {
            express: true,
            socketio: true,
            peerjs: true
        }
```

```
    });
});

// API des statistiques
app.get('/api/stats', (req, res) => {
    const stats = roomManager.getStats();
    res.json({
        server: {
            uptime: process.uptime(),
            memory: process.memoryUsage(),
            ssl: true
        },
        rooms: stats,
        socketio: {
            connected: io.engine.clientsCount
        }
    });
});

// API des salles
app.get('/api/rooms', (req, res) => {
    const rooms = roomManager.getAllRooms();
    res.json({
        success: true,
        rooms: rooms
    });
});

app.post('/api/rooms', (req, res) => {
    try {
        const { roomId, maxClients, metadata } = req.body;
        const room = roomManager.createRoom(roomId, maxClients, metadata);

        res.status(201).json({
            success: true,
            room: roomManager.getRoomInfo(room.id)
        });
    } catch (error) {
        res.status(400).json({
            success: false,
            error: error.message
        });
    }
});

// API de test
app.get('/api/test', (req, res) => {
    res.json({
        message: 'Application WebRTC complète fonctionnelle !',
```

```
        timestamp: new Date().toISOString(),
        ssl: true,
        endpoints: [
            'GET /',
            'GET /api/health',
            'GET /api/stats',
            'GET /api/rooms',
            'POST /api/rooms',
            'GET /api/test'
        ]
    });
});

// Gestion des erreurs 404
app.use((req, res) => {
    res.status(404).json({
        error: 'Endpoint non trouvé',
        path: req.path
    });
});

// Gestion globale des erreurs
app.use((error, req, res, next) => {
    console.error('Erreur serveur:', error);
    res.status(500).json({
        error: 'Erreur interne du serveur'
    });
});

// ===== ÉVÉNEMENTS PEERJS =====

peerServer.on('connection', (client) => {
    console.log(`[PEERJS] Client PeerJS connecté: ${client.getId()}`);
});

peerServer.on('disconnect', (client) => {
    console.log(`[PEERJS] Client PeerJS déconnecté: ${client.getId()}`);
});

// ===== DÉMARRAGE DU SERVEUR =====

const PORT = config.server.port;

server.listen(PORT, () => {
    console.log('\n🚀 Application WebRTC complète démarrée !');
    console.log(`🔒 URL HTTPS: https://localhost:${PORT}/`);
    console.log(`📡 Socket.IO: wss://localhost:${PORT}/socket.io/`);
    console.log(`🔗 PeerJS: https://localhost:${PORT}${config.peerjs.path}`);
    console.log(`📊 API: https://localhost:${PORT}/api/`);
```

```
    console.log('\n🔐 IMPORTANT: Acceptez le certificat auto-signé dans votre
navigateur');
    console.log('✋ Appuyez sur CTRL+C pour arrêter\n');
});

// Gestion propre de l'arrêt
process.on('SIGINT', () => {
    console.log('\n🛑 Arrêt de l\'application...');
    server.close(() => {
        console.log('✅ Serveur fermé proprement');
        process.exit(0);
    });
});

process.on('SIGTERM', () => {
    console.log('🔄 Signal SIGTERM reçu, arrêt en cours...');
    server.close(() => {
        process.exit(0);
    });
});

module.exports = { app, server, io, roomManager };
```

4. Interface Client Complète

Le fichier HTML principal (**public/index.html**)

nano public/index.html

```
root@machine1:/home/sergio/webrtc-complete-app# nano public/index.html
root@machine1:/home/sergio/webrtc-complete-app#
```

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>WebRTC Complete App - Conférence Vidéo Sécurisée</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="app-container">
        <!-- Header avec état de connexion -->
        <header class="app-header">
            <div class="header-content">
                <h1>📹 WebRTC Conference</h1>
                <div class="connection-status">
                    <div class="status-indicator" id="connection-status">
```

```
                    <span class="status-dot offline"></span>
                    <span class="status-text">Connexion...</span>
                </div>
                <div class="ssl-indicator">
                    <span class="ssl-icon">🔒</span>
                    <span class="ssl-text">HTTPS Sécurisé</span>
                </div>
            </div>
        </div>
    </header>

    <!-- Section principale -->
    <main class="app-main">
        <!-- Panneau de contrôle -->
        <aside class="control-panel">
            <div class="panel-section">
                <h3>🏠 Gestion des Salles</h3>
                <div class="room-controls">
                    <input type="text" id="room-id" placeholder="Nom de la
salle">
                    <input type="text" id="nickname" placeholder="Votre
nom">
                    <button id="create-room-btn" class="btn btn-
primary">Créer</button>
                    <button id="join-room-btn" class="btn btn-
success">Rejoindre</button>
                    <button id="leave-room-btn" class="btn btn-danger"
disabled>Quitter</button>
                </div>

                <div class="room-info" id="room-info" style="display:
none;">
                    <h4>📋 Informations de la Salle</h4>
                    <div class="info-item">
                        <span class="label">Nom:</span>
                        <span class="value" id="current-room-name"></span>
                    </div>
                    <div class="info-item">
                        <span class="label">Participants:</span>
                        <span class="value" id="current-room-
clients"></span>
                    </div>
                    <div class="info-item">
                        <span class="label">ID:</span>
                        <span class="value" id="current-room-id"></span>
                    </div>
                </div>
            </div>
```

```
                <div class="panel-section">
                    <h3>👥 Participants</h3>
                    <div class="participants-list" id="participants-list">
                        <p class="no-participants">Aucun participant</p>
                    </div>
                </div>

                <div class="panel-section">
                    <h3>🎮 Contrôles</h3>
                    <div class="media-controls">
                        <button id="toggle-video" class="btn btn-
icon">📹</button>
                        <button id="toggle-audio" class="btn btn-
icon">🎤</button>
                        <button id="share-screen" class="btn btn-
icon">🖥️</button>
                        <button id="start-call" class="btn btn-success"
disabled>📞 Appeler</button>
                        <button id="end-call" class="btn btn-danger"
disabled>📵 Raccrocher</button>
                    </div>
                </div>
            </aside>

            <!-- Zone vidéo et chat -->
            <section class="content-area">
                <!-- Zone vidéo -->
                <div class="video-area">
                    <div class="video-grid" id="video-grid">
                        <div class="video-container local-video">
                            <video id="local-video" autoplay muted
playsinline></video>
                            <div class="video-label">Vous</div>
                            <div class="video-controls">
                                <span class="recording-indicator"
id="recording-indicator" style="display: none;">⏺ REC</span>
                            </div>
                        </div>
                        <div class="video-container remote-video"
style="display: none;">
                            <video id="remote-video" autoplay
playsinline></video>
                            <div class="video-label">Participant</div>
                        </div>
                    </div>
                </div>

                <!-- Zone chat -->
                <div class="chat-area">
```

```
                    <div class="chat-header">
                        <h3>💬 Chat</h3>
                        <button id="clear-chat" class="btn btn-
small">Effacer</button>
                    </div>
                    <div class="chat-messages" id="chat-messages"></div>
                    <div class="chat-input">
                        <input type="text" id="message-input"
placeholder="Tapez votre message...">
                        <button id="send-message" class="btn btn-
primary">Envoyer</button>
                    </div>
                    <div class="typing-indicator" id="typing-indicator"
style="display: none;">
                        <span class="typing-text">Quelqu'un écrit...</span>
                    </div>
                </div>
            </section>
        </main>

        <!-- Notifications -->
        <div class="notifications" id="notifications"></div>

        <!-- Logs de debug (masqués par défaut) -->
        <div class="debug-panel" id="debug-panel" style="display: none;">
            <div class="debug-header">
                <h4>🔧 Logs de Debug</h4>
                <button id="toggle-debug" class="btn btn-
small">Masquer</button>
            </div>
            <div class="debug-logs" id="debug-logs"></div>
        </div>
    </div>

    <!-- Scripts -->
    <script src="https://unpkg.com/socket.io@4.7.2/client-
dist/socket.io.js"></script>
    <script src="https://unpkg.com/peerjs@1.4.7/dist/peerjs.min.js"></script>
    <script src="js/app.js"></script>

    <!-- Bouton de debug flottant -->
    <button id="debug-toggle" class="debug-toggle"
onclick="toggleDebugPanel()">🔧</button>
</body>
</html>
```

- Création de fichiers CSS principal (**public/css/style.css**)

nano public/css/style.css

```
root@machine1:/home/sergio/webrtc-complete-app# nano public/css/style.css
root@machine1:/home/sergio/webrtc-complete-app#
```

```
/* ================================
   APPLICATION WEBRTC COMPLÈTE - STYLES
   ================================ */

/* Reset et base */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    color: #333;
    overflow-x: hidden;
}

.app-container {
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}

/* ================= HEADER ================= */

.app-header {
    background: rgba(255, 255, 255, 0.95);
    backdrop-filter: blur(10px);
    border-bottom: 1px solid rgba(255, 255, 255, 0.2);
    padding: 15px 20px;
    box-shadow: 0 2px 20px rgba(0, 0, 0, 0.1);
}

.header-content {
    display: flex;
    justify-content: space-between;
    align-items: center;
    max-width: 1400px;
    margin: 0 auto;
```

```
}

.app-header h1 {
    color: #4a5568;
    font-size: 1.8rem;
    font-weight: 600;
}

.connection-status {
    display: flex;
    align-items: center;
    gap: 20px;
}

.status-indicator {
    display: flex;
    align-items: center;
    gap: 8px;
    padding: 8px 12px;
    background: rgba(255, 255, 255, 0.8);
    border-radius: 20px;
    font-size: 0.9rem;
    font-weight: 500;
}

.status-dot {
    width: 8px;
    height: 8px;
    border-radius: 50%;
    animation: pulse 2s infinite;
}

.status-dot.online { background: #48bb78; }
.status-dot.offline { background: #f56565; }
.status-dot.connecting { background: #ed8936; }

.ssl-indicator {
    display: flex;
    align-items: center;
    gap: 5px;
    color: #38b2ac;
    font-size: 0.85rem;
    font-weight: 600;
}

@keyframes pulse {
    0%, 100% { opacity: 1; }
    50% { opacity: 0.5; }
}
```

```
/* ================== MAIN LAYOUT ================== */

.app-main {
    display: flex;
    flex: 1;
    max-width: 1400px;
    margin: 0 auto;
    padding: 20px;
    gap: 20px;
}

/* ================== PANEL DE CONTRÔLE ================== */

.control-panel {
    width: 300px;
    background: rgba(255, 255, 255, 0.95);
    border-radius: 15px;
    padding: 20px;
    height: fit-content;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
}

.panel-section {
    margin-bottom: 25px;
    padding-bottom: 20px;
    border-bottom: 1px solid #e2e8f0;
}

.panel-section:last-child {
    border-bottom: none;
    margin-bottom: 0;
}

.panel-section h3 {
    color: #4a5568;
    font-size: 1.1rem;
    margin-bottom: 15px;
    font-weight: 600;
}

.room-controls {
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.room-controls input {
    padding: 12px;
```

```
    border: 2px solid #e2e8f0;
    border-radius: 8px;
    font-size: 0.9rem;
    transition: border-color 0.3s ease;
}

.room-controls input:focus {
    outline: none;
    border-color: #4299e1;
}

.room-info {
    background: #f7fafc;
    padding: 15px;
    border-radius: 8px;
    margin-top: 15px;
}

.room-info h4 {
    color: #4a5568;
    margin-bottom: 10px;
    font-size: 0.95rem;
}

.info-item {
    display: flex;
    justify-content: space-between;
    margin-bottom: 8px;
    font-size: 0.85rem;
}

.info-item .label {
    font-weight: 600;
    color: #718096;
}

.info-item .value {
    color: #4a5568;
    font-family: monospace;
}

.participants-list {
    max-height: 200px;
    overflow-y: auto;
}

.participant-item {
    display: flex;
    align-items: center;
```

```
    gap: 10px;
    padding: 8px;
    margin-bottom: 5px;
    background: #f7fafc;
    border-radius: 8px;
    font-size: 0.85rem;
}

.participant-status {
    width: 6px;
    height: 6px;
    border-radius: 50%;
    background: #48bb78;
}

.no-participants {
    color: #a0aec0;
    font-style: italic;
    text-align: center;
    padding: 20px;
    font-size: 0.9rem;
}

.media-controls {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 8px;
    margin-bottom: 15px;
}

.media-controls .btn:nth-child(4),
.media-controls .btn:nth-child(5) {
    grid-column: span 3;
}

/* ================== BOUTONS ================== */

.btn {
    padding: 10px 15px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-weight: 600;
    font-size: 0.85rem;
    transition: all 0.3s ease;
    text-align: center;
    text-decoration: none;
    display: inline-flex;
    align-items: center;
```

```
    justify-content: center;
    gap: 5px;
}

.btn:disabled {
    opacity: 0.5;
    cursor: not-allowed;
}

.btn-primary { background: #4299e1; color: white; }
.btn-success { background: #48bb78; color: white; }
.btn-danger { background: #f56565; color: white; }
.btn-warning { background: #ed8936; color: white; }
.btn-info { background: #38b2ac; color: white; }
.btn-small { padding: 6px 12px; font-size: 0.8rem; }
.btn-icon { padding: 8px; font-size: 1.2rem; }

.btn-primary:hover { background: #3182ce; }
.btn-success:hover { background: #38a169; }
.btn-danger:hover { background: #e53e3e; }
.btn-warning:hover { background: #dd6b20; }
.btn-info:hover { background: #319795; }

/* ================= ZONE CONTENU ================= */

.content-area {
    flex: 1;
    display: flex;
    flex-direction: column;
    gap: 20px;
}

/* ================= ZONE VIDÉO ================= */

.video-area {
    background: rgba(255, 255, 255, 0.95);
    border-radius: 15px;
    padding: 20px;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
}

.video-grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
    min-height: 400px;
}

.video-container {
```

```
    position: relative;
    background: #000;
    border-radius: 12px;
    overflow: hidden;
    aspect-ratio: 16/9;
}

.video-container video {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.video-label {
    position: absolute;
    bottom: 10px;
    left: 10px;
    background: rgba(0, 0, 0, 0.7);
    color: white;
    padding: 5px 10px;
    border-radius: 15px;
    font-size: 0.8rem;
    font-weight: 500;
}

.video-controls {
    position: absolute;
    top: 10px;
    right: 10px;
    display: flex;
    gap: 5px;
}

.recording-indicator {
    background: rgba(229, 62, 62, 0.9);
    color: white;
    padding: 4px 8px;
    border-radius: 12px;
    font-size: 0.7rem;
    font-weight: bold;
    animation: blink 1s infinite;
}

@keyframes blink {
    0%, 100% { opacity: 1; }
    50% { opacity: 0.5; }
}

/* ================== ZONE CHAT ================== */
```

```css
.chat-area {
    background: rgba(255, 255, 255, 0.95);
    border-radius: 15px;
    padding: 20px;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
    height: 400px;
}

.chat-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 15px;
    padding-bottom: 10px;
    border-bottom: 2px solid #e2e8f0;
}

.chat-header h3 {
    color: #4a5568;
    font-size: 1.1rem;
}

.chat-messages {
    flex: 1;
    overflow-y: auto;
    padding: 10px;
    background: #f7fafc;
    border-radius: 8px;
    margin-bottom: 15px;
    border: 1px solid #e2e8f0;
}

.chat-input {
    display: flex;
    gap: 10px;
}

.chat-input input {
    flex: 1;
    padding: 12px;
    border: 2px solid #e2e8f0;
    border-radius: 8px;
    font-size: 0.9rem;
}

.chat-input input:focus {
```

```
    outline: none;
    border-color: #4299e1;
}

.message {
    margin-bottom: 10px;
    padding: 10px 12px;
    border-radius: 12px;
    max-width: 80%;
    word-wrap: break-word;
    animation: slideIn 0.3s ease;
}

.message.own {
    background: #4299e1;
    color: white;
    margin-left: auto;
    text-align: right;
}

.message.other {
    background: #e2e8f0;
    color: #333;
}

.message-time {
    font-size: 0.7rem;
    opacity: 0.7;
    margin-top: 5px;
}

.typing-indicator {
    color: #718096;
    font-style: italic;
    font-size: 0.85rem;
    padding: 5px 10px;
}

@keyframes slideIn {
    from { opacity: 0; transform: translateY(10px); }
    to { opacity: 1; transform: translateY(0); }
}

/* ================== NOTIFICATIONS ================== */

.notifications {
    position: fixed;
    top: 20px;
    right: 20px;
```

```
    z-index: 1000;
    display: flex;
    flex-direction: column;
    gap: 10px;
}

.notification {
    background: rgba(255, 255, 255, 0.95);
    border-left: 4px solid #4299e1;
    padding: 15px;
    border-radius: 8px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
    max-width: 300px;
    animation: slideInRight 0.3s ease;
}

.notification.success { border-left-color: #48bb78; }
.notification.error { border-left-color: #f56565; }
.notification.warning { border-left-color: #ed8936; }

@keyframes slideInRight {
    from { opacity: 0; transform: translateX(100%); }
    to { opacity: 1; transform: translateX(0); }
}

/* ================== DEBUG PANEL ================== */

.debug-panel {
    position: fixed;
    bottom: 20px;
    left: 20px;
    right: 20px;
    max-height: 300px;
    background: rgba(0, 0, 0, 0.9);
    color: #00ff00;
    border-radius: 8px;
    overflow: hidden;
    font-family: 'Courier New', monospace;
    z-index: 1000;
}

.debug-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 15px;
    background: rgba(255, 255, 255, 0.1);
    border-bottom: 1px solid rgba(255, 255, 255, 0.2);
}
```

```css
.debug-header h4 {
    color: #00ff00;
    font-size: 0.9rem;
}

.debug-logs {
    padding: 15px;
    max-height: 250px;
    overflow-y: auto;
    font-size: 0.8rem;
    line-height: 1.4;
}

.debug-toggle {
    position: fixed;
    bottom: 20px;
    right: 20px;
    width: 50px;
    height: 50px;
    border-radius: 50%;
    background: #4299e1;
    color: white;
    border: none;
    font-size: 1.5rem;
    cursor: pointer;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
    transition: all 0.3s ease;
    z-index: 1001;
}

.debug-toggle:hover {
    background: #3182ce;
    transform: scale(1.1);
}

/* ================== RESPONSIVE ================== */

@media (max-width: 1024px) {
    .app-main {
        flex-direction: column;
    }

    .control-panel {
        width: 100%;
        order: 2;
    }

    .content-area {
```

```
        order: 1;
    }

    .video-grid {
        grid-template-columns: 1fr;
    }
}

@media (max-width: 768px) {
    .header-content {
        flex-direction: column;
        gap: 15px;
        text-align: center;
    }

    .connection-status {
        justify-content: center;
    }

    .room-controls {
        gap: 8px;
    }

    .media-controls {
        grid-template-columns: repeat(2, 1fr);
    }

    .chat-area {
        height: 300px;
    }

    .notifications {
        left: 10px;
        right: 10px;
    }

    .debug-panel {
        left: 10px;
        right: 10px;
    }
}

/* ================= ANIMATIONS ================= */

.fade-in {
    animation: fadeIn 0.5s ease;
}

.slide-up {
```

```
    animation: slideUp 0.3s ease;
}

@keyframes fadeIn {
    from { opacity: 0; }
    to { opacity: 1; }
}

@keyframes slideUp {
    from { opacity: 0; transform: translateY(20px); }
    to { opacity: 1; transform: translateY(0); }
}

/* ================== SCROLLBAR CUSTOM ================== */

.chat-messages::-webkit-scrollbar,
.debug-logs::-webkit-scrollbar,
.participants-list::-webkit-scrollbar {
    width: 6px;
}

.chat-messages::-webkit-scrollbar-track,
.debug-logs::-webkit-scrollbar-track,
.participants-list::-webkit-scrollbar-track {
    background: #f1f1f1;
    border-radius: 3px;
}

.chat-messages::-webkit-scrollbar-thumb,
.debug-logs::-webkit-scrollbar-thumb,
.participants-list::-webkit-scrollbar-thumb {
    background: #cbd5e0;
    border-radius: 3px;
}

.chat-messages::-webkit-scrollbar-thumb:hover,
.debug-logs::-webkit-scrollbar-thumb:hover,
.participants-list::-webkit-scrollbar-thumb:hover {
    background: #a0aec0;
}
```

- Le fichier **JavaScript** principal (**public/js/app.js**)

nano public/js/app.js

```
root@machine1:/home/sergio/webrtc-complete-app# nano public/js/app.js
root@machine1:/home/sergio/webrtc-complete-app#
```

```
/**
 * APPLICATION WEBRTC COMPLÈTE
 * Intégration PeerJS + Socket.IO + WebRTC
 */

class WebRTCCompleteApp {
    constructor() {
        this.socket = null;
        this.peer = null;
        this.localStream = null;
        this.remoteStreams = new Map();
        this.currentRoom = null;
        this.isVideoEnabled = true;
        this.isAudioEnabled = true;
        this.isScreenSharing = false;
        this.peers = new Map(); // Connexions peer actives

        this.initializeElements();
        this.initializeSocket();
        this.initializePeer();
        this.setupEventListeners();
        this.initializeLocalMedia();

        this.log('Application WebRTC complète initialisée', 'info');
    }

    initializeElements() {
        this.elements = {
            // Status
            connectionStatus: document.getElementById('connection-status'),

            // Room controls
            roomIdInput: document.getElementById('room-id'),
            nicknameInput: document.getElementById('nickname'),
            createRoomBtn: document.getElementById('create-room-btn'),
            joinRoomBtn: document.getElementById('join-room-btn'),
            leaveRoomBtn: document.getElementById('leave-room-btn'),

            // Room info
            roomInfo: document.getElementById('room-info'),
            currentRoomName: document.getElementById('current-room-name'),
            currentRoomClients: document.getElementById('current-room-
clients'),
            currentRoomId: document.getElementById('current-room-id'),

            // Participants
            participantsList: document.getElementById('participants-list'),

            // Media controls
```

```
            toggleVideo: document.getElementById('toggle-video'),
            toggleAudio: document.getElementById('toggle-audio'),
            shareScreen: document.getElementById('share-screen'),
            startCall: document.getElementById('start-call'),
            endCall: document.getElementById('end-call'),

            // Video
            localVideo: document.getElementById('local-video'),
            remoteVideo: document.getElementById('remote-video'),
            videoGrid: document.getElementById('video-grid'),
            recordingIndicator: document.getElementById('recording-
indicator'),

            // Chat
            chatMessages: document.getElementById('chat-messages'),
            messageInput: document.getElementById('message-input'),
            sendMessage: document.getElementById('send-message'),
            clearChat: document.getElementById('clear-chat'),
            typingIndicator: document.getElementById('typing-indicator'),

            // Debug
            debugPanel: document.getElementById('debug-panel'),
            debugLogs: document.getElementById('debug-logs'),
            notifications: document.getElementById('notifications')
        };
    }

    initializeSocket() {
        // Connexion Socket.IO sécurisée
        this.socket = io({
            secure: true,
            rejectUnauthorized: false // Pour les certificats auto-signés
        });

        // Événements de connexion
        this.socket.on('connect', () => {
            this.log('Connecté au serveur Socket.IO', 'success');
            this.updateConnectionStatus('online', 'Connecté');
        });

        this.socket.on('disconnect', () => {
            this.log('Déconnecté du serveur Socket.IO', 'warning');
            this.updateConnectionStatus('offline', 'Déconnecté');
        });

        this.socket.on('connect_error', (error) => {
            this.log(`Erreur de connexion Socket.IO: ${error.message}`,
'error');
            this.updateConnectionStatus('offline', 'Erreur de connexion');
```

```
        });

        // Événements de salle
        this.socket.on('room-created', (room) => {
            this.log(`Salle créée: ${room.metadata.name}`, 'info');
        });

        this.socket.on('client-joined', (data) => {
            this.log(`${data.client.info.nickname} a rejoint la salle`,
'info');
            this.updateParticipantsList();
            this.showNotification(`${data.client.info.nickname} a rejoint la
salle`, 'success');
        });

        this.socket.on('client-left', (data) => {
            this.log(`Client ${data.clientId} a quitté la salle`, 'info');
            this.updateParticipantsList();
            this.showNotification('Un participant a quitté la salle', 'info');
        });

        // Événements WebRTC
        this.socket.on('webrtc-offer', (data) => {
            this.handleWebRTCOffer(data);
        });

        this.socket.on('webrtc-answer', (data) => {
            this.handleWebRTCAnswer(data);
        });

        this.socket.on('webrtc-ice-candidate', (data) => {
            this.handleICECandidate(data);
        });

        this.socket.on('call-incoming', (data) => {
            this.handleIncomingCall(data);
        });

        this.socket.on('call-ended', (data) => {
            this.handleCallEnded(data);
        });

        // Événements de chat
        this.socket.on('chat-message', (data) => {
            this.displayChatMessage(data);
        });

        this.socket.on('typing-start', (data) => {
            this.showTypingIndicator(data.from);
```

```
        });

        this.socket.on('typing-stop', (data) => {
            this.hideTypingIndicator();
        });
    }

    initializePeer() {
        // Connexion PeerJS sécurisée
        this.peer = new Peer({
            host: 'localhost',
            port: 8443,
            path: '/peerjs',
            secure: true,
            config: {
                iceServers: [
                    { urls: 'stun:stun.l.google.com:19302' },
                    { urls: 'stun:stun1.l.google.com:19302' }
                ]
            }
        });

        this.peer.on('open', (id) => {
            this.peerId = id;
            this.log(`PeerJS connecté avec l'ID: ${id}`, 'success');
        });

        this.peer.on('error', (error) => {
            this.log(`Erreur PeerJS: ${error.message}`, 'error');
        });

        this.peer.on('call', (call) => {
            this.log('Appel PeerJS entrant', 'info');
            if (this.localStream) {
                call.answer(this.localStream);
                this.setupPeerCall(call);
            }
        });
    }

    setupEventListeners() {
        // Room controls
        this.elements.createRoomBtn.addEventListener('click', () =>
this.createRoom());
        this.elements.joinRoomBtn.addEventListener('click', () =>
this.joinRoom());
        this.elements.leaveRoomBtn.addEventListener('click', () =>
this.leaveRoom());
```

```
        // Media controls
        this.elements.toggleVideo.addEventListener('click', () =>
this.toggleVideo());
        this.elements.toggleAudio.addEventListener('click', () =>
this.toggleAudio());
        this.elements.shareScreen.addEventListener('click', () =>
this.toggleScreenShare());
        this.elements.startCall.addEventListener('click', () =>
this.startCall());
        this.elements.endCall.addEventListener('click', () => this.endCall());

        // Chat
        this.elements.sendMessage.addEventListener('click', () =>
this.sendMessage());
        this.elements.messageInput.addEventListener('keypress', (e) => {
            if (e.key === 'Enter') {
                this.sendMessage();
            } else {
                this.handleTyping();
            }
        });
        this.elements.clearChat.addEventListener('click', () =>
this.clearChat());

        // Entrées clavier
        this.elements.roomIdInput.addEventListener('keypress', (e) => {
            if (e.key === 'Enter') this.joinRoom();
        });

        this.elements.nicknameInput.addEventListener('keypress', (e) => {
            if (e.key === 'Enter') this.joinRoom();
        });
    }

    async initializeLocalMedia() {
        try {
            this.localStream = await navigator.mediaDevices.getUserMedia({
                video: { width: 1280, height: 720 },
                audio: true
            });

            this.elements.localVideo.srcObject = this.localStream;
            this.log('Médias locaux initialisés', 'success');

        } catch (error) {
            this.log(`Erreur d'accès aux médias: ${error.message}`, 'error');
            this.showNotification('Impossible d\'accéder à la
caméra/microphone', 'error');
        }
```

```
    }

    // ===== GESTION DES SALLES =====

    createRoom() {
        const roomId = this.elements.roomIdInput.value.trim();
        const nickname = this.elements.nicknameInput.value.trim() ||
'Anonyme';

        if (!roomId) {
            this.showNotification('Veuillez entrer un nom de salle',
'warning');
            return;
        }

        this.socket.emit('create-room', {
            roomId: roomId,
            maxClients: 10,
            metadata: {
                name: roomId,
                description: 'Salle de conférence vidéo'
            }
        }, (response) => {
            if (response.success) {
                this.log(`Salle créée: ${roomId}`, 'success');
                this.joinRoom(); // Rejoindre automatiquement la salle créée
            } else {
                this.log(`Erreur de création: ${response.error}`, 'error');
                this.showNotification(response.error, 'error');
            }
        });
    }

    joinRoom() {
        const roomId = this.elements.roomIdInput.value.trim();
        const nickname = this.elements.nicknameInput.value.trim() ||
'Anonyme';

        if (!roomId) {
            this.showNotification('Veuillez entrer un nom de salle',
'warning');
            return;
        }

        this.socket.emit('join-room', {
            roomId: roomId,
            clientInfo: {
                nickname: nickname,
                peerId: this.peerId,
```

```
                userAgent: navigator.userAgent
            }
        }, (response) => {
            if (response.success) {
                this.currentRoom = response.room;
                this.updateRoomUI();
                this.log(`Rejoint la salle: ${roomId}`, 'success');
                this.showNotification(`Bienvenue dans la salle ${roomId}`,
'success');
            } else {
                this.log(`Erreur pour rejoindre: ${response.error}`, 'error');
                this.showNotification(response.error, 'error');
            }
        });
    }

    leaveRoom() {
        if (!this.currentRoom) return;

        this.socket.emit('leave-room', (response) => {
            if (response.success) {
                this.currentRoom = null;
                this.updateRoomUI();
                this.log('Salle quittée', 'info');
                this.showNotification('Vous avez quitté la salle', 'info');
            }
        });
    }

    updateRoomUI() {
        if (this.currentRoom) {
            this.elements.roomInfo.style.display = 'block';
            this.elements.currentRoomName.textContent =
this.currentRoom.metadata.name;
            this.elements.currentRoomClients.textContent =
`${this.currentRoom.clientCount}/${this.currentRoom.maxClients}`;
            this.elements.currentRoomId.textContent = this.currentRoom.id;

            this.elements.leaveRoomBtn.disabled = false;
            this.elements.startCall.disabled = false;

            this.updateParticipantsList();
        } else {
            this.elements.roomInfo.style.display = 'none';
            this.elements.leaveRoomBtn.disabled = true;
            this.elements.startCall.disabled = true;
            this.elements.endCall.disabled = true;

            this.clearParticipantsList();
```

```
        }
    }

    updateParticipantsList() {
        if (!this.currentRoom) return;

        this.socket.emit('get-room-info', (response) => {
            if (response.success && response.room) {
                const participants = response.room.clients || [];
                this.elements.participantsList.innerHTML = '';

                if (participants.length === 0) {
                    this.elements.participantsList.innerHTML = '<p class="no-
participants">Aucun participant</p>';
                } else {
                    participants.forEach(participant => {
                        const item = document.createElement('div');
                        item.className = 'participant-item';
                        item.innerHTML = `
                            <div class="participant-status"></div>
                            <span>${participant.nickname ||
participant.id.substring(0, 8)}</span>
                        `;
                        this.elements.participantsList.appendChild(item);
                    });
                }
            }
        });
    }

    clearParticipantsList() {
        this.elements.participantsList.innerHTML = '<p class="no-
participants">Aucun participant</p>';
    }

    // ===== CONTRÔLES MÉDIAS =====

    toggleVideo() {
        if (!this.localStream) return;

        const videoTrack = this.localStream.getVideoTracks()[0];
        if (videoTrack) {
            this.isVideoEnabled = !this.isVideoEnabled;
            videoTrack.enabled = this.isVideoEnabled;

            this.elements.toggleVideo.textContent = this.isVideoEnabled ? '📹'
: '📹❌';
            this.log(`Vidéo ${this.isVideoEnabled ? 'activée' :
'désactivée'}`, 'info');
```

```
        }
    }

    toggleAudio() {
        if (!this.localStream) return;

        const audioTrack = this.localStream.getAudioTracks()[0];
        if (audioTrack) {
            this.isAudioEnabled = !this.isAudioEnabled;
            audioTrack.enabled = this.isAudioEnabled;

            this.elements.toggleAudio.textContent = this.isAudioEnabled ? '🎤' 
: '🎤❌';
            this.log(`Audio ${this.isAudioEnabled ? 'activé' : 'désactivé'}`, 
'info');
        }
    }

    async toggleScreenShare() {
        if (!this.isScreenSharing) {
            try {
                const screenStream = await 
navigator.mediaDevices.getDisplayMedia({
                    video: true,
                    audio: true
                });

                // Remplacer le stream local
                this.elements.localVideo.srcObject = screenStream;
                this.isScreenSharing = true;
                this.elements.shareScreen.textContent = '📺 Stop';

                this.log('Partage d\'écran démarré', 'success');
                this.socket.emit('screen-share-start');

                // Arrêter le partage quand l'utilisateur ferme la fenêtre
                screenStream.getVideoTracks()[0].onended = () => {
                    this.stopScreenShare();
                };

            } catch (error) {
                this.log(`Erreur de partage d'écran: ${error.message}`, 
'error');
            }
        } else {
            this.stopScreenShare();
        }
    }
```

```
    stopScreenShare() {
        this.elements.localVideo.srcObject = this.localStream;
        this.isScreenSharing = false;
        this.elements.shareScreen.textContent = '▣ □';
        this.log('Partage d\'écran arrêté', 'info');
        this.socket.emit('screen-share-stop');
    }

    // ===== APPELS WEBRTC =====

    startCall() {
        if (!this.currentRoom) return;

        // Démarrer un appel avec tous les participants de la salle
        this.socket.emit('start-call', { roomId: this.currentRoom.id });
        this.elements.startCall.disabled = true;
        this.elements.endCall.disabled = false;
        this.log('Appel démarré', 'success');
    }

    endCall() {
        this.socket.emit('end-call', { roomId: this.currentRoom.id });
        this.elements.startCall.disabled = false;
        this.elements.endCall.disabled = true;

        // Fermer toutes les connexions peer
        this.peers.forEach(peer => peer.close());
        this.peers.clear();

        this.elements.remoteVideo.srcObject = null;
        this.elements.remoteVideo.parentElement.style.display = 'none';

        this.log('Appel terminé', 'info');
    }

    handleIncomingCall(data) {
        this.log(`Appel entrant de ${data.from}`, 'info');

        if (confirm('Appel entrant. Accepter ?')) {
            // Répondre à l'appel via PeerJS
            if (this.peer && this.localStream) {
                const call = this.peer.call(data.from, this.localStream);
                this.setupPeerCall(call);
            }
        }
    }

    handleCallEnded(data) {
        this.log(`Appel terminé par ${data.from}`, 'info');
```

```
        this.endCall();
    }

    setupPeerCall(call) {
        call.on('stream', (remoteStream) => {
            this.elements.remoteVideo.srcObject = remoteStream;
            this.elements.remoteVideo.parentElement.style.display = 'block';
            this.log('Stream distant reçu', 'success');
        });

        call.on('close', () => {
            this.elements.remoteVideo.srcObject = null;
            this.elements.remoteVideo.parentElement.style.display = 'none';
            this.log('Connexion peer fermée', 'info');
        });

        this.peers.set(call.peer, call);
    }

    // ===== CHAT =====

    sendMessage() {
        const message = this.elements.messageInput.value.trim();
        if (!message || !this.currentRoom) return;

        this.socket.emit('chat-message', {
            message: message,
            timestamp: new Date().toISOString()
        });

        this.elements.messageInput.value = '';
        this.hideTypingIndicator();
    }

    displayChatMessage(data) {
        const messageDiv = document.createElement('div');
        messageDiv.className = `message ${data.from === this.socket.id ? 'own'
: 'other'}`;

        const time = new Date(data.timestamp).toLocaleTimeString();
        messageDiv.innerHTML = `
            <div>${data.message}</div>
            <div class="message-time">${time}</div>
        `;

        this.elements.chatMessages.appendChild(messageDiv);
        this.elements.chatMessages.scrollTop =
this.elements.chatMessages.scrollHeight;
    }
```

```
    handleTyping() {
        if (!this.typingTimeout) {
            this.socket.emit('typing-start');
        }

        clearTimeout(this.typingTimeout);
        this.typingTimeout = setTimeout(() => {
            this.socket.emit('typing-stop');
            this.typingTimeout = null;
        }, 3000);
    }

    showTypingIndicator(from) {
        this.elements.typingIndicator.style.display = 'block';
        this.elements.typingIndicator.textContent = `${from} écrit...`;
    }

    hideTypingIndicator() {
        this.elements.typingIndicator.style.display = 'none';
    }

    clearChat() {
        this.elements.chatMessages.innerHTML = '';
        this.log('Chat effacé', 'info');
    }

    // ===== UTILITAIRES =====

    updateConnectionStatus(status, text) {
        const statusElement = this.elements.connectionStatus;
        const dot = statusElement.querySelector('.status-dot');
        const textElement = statusElement.querySelector('.status-text');

        dot.className = `status-dot ${status}`;
        textElement.textContent = text;
    }

    showNotification(message, type = 'info') {
        const notification = document.createElement('div');
        notification.className = `notification ${type}`;
        notification.textContent = message;

        this.elements.notifications.appendChild(notification);

        setTimeout(() => {
            notification.remove();
        }, 5000);
    }
```

```
    log(message, type = 'info') {
        const timestamp = new Date().toLocaleTimeString();
        const logEntry = document.createElement('div');
        logEntry.innerHTML = `<span style="color: #666">[${timestamp}]</span>
<span style="color: ${this.getLogColor(type)}">[${type.toUpperCase()}]</span>
${message}`;

        this.elements.debugLogs.appendChild(logEntry);
        this.elements.debugLogs.scrollTop =
this.elements.debugLogs.scrollHeight;

        console.log(`[WebRTC] [${type.toUpperCase()}] ${message}`);
    }

    getLogColor(type) {
        const colors = {
            info: '#00ff00',
            success: '#00ff88',
            warning: '#ffaa00',
            error: '#ff4444'
        };
        return colors[type] || '#00ff00';
    }
}

// ===== FONCTIONS GLOBALES =====

function toggleDebugPanel() {
    const panel = document.getElementById('debug-panel');
    panel.style.display = panel.style.display === 'none' ? 'block' : 'none';
}

// ===== INITIALISATION =====

document.addEventListener('DOMContentLoaded', () => {
    window.webrtcApp = new WebRTCCompleteApp();
});
```

5. Scripts de Démarrage et Test

Script de démarrage (**scripts/start-app.sh**)

nano scripts/start-app.sh

```
root@machine1:/home/sergio/webrtc-complete-app# nano scripts/start-app.sh
root@machine1:/home/sergio/webrtc-complete-app#
```

```
#!/bin/bash
```

```
echo "=== Démarrage de l'Application WebRTC Complète ==="

cd /home/sergio/webrtc-complete-app

# Fonction de nettoyage
cleanup() {
    echo "🛑 Arrêt de l'application..."
    pkill -f "node.*app.js" 2>/dev/null || true
    exit 0
}

trap cleanup SIGINT SIGTERM

# Vérifier la structure du projet
if [ ! -f "server/app.js" ]; then
    echo "❌ Fichier server/app.js non trouvé"
    exit 1
fi

# Vérifier les certificats SSL
if [ ! -f "ssl/server-cert.pem" ] || [ ! -f "ssl/server-key.pem" ]; then
    echo "🔐 Génération des certificats SSL..."
    node scripts/generate-ssl.js
fi

# Installer les dépendances si nécessaire
if [ ! -d "node_modules" ]; then
    echo "📦 Installation des dépendances..."
    npm install
fi

# Créer les dossiers nécessaires
mkdir -p logs public/{css,js,assets}

echo "🚀 Démarrage de l'application..."
echo "🔒 L'application sera accessible sur https://localhost:8443"
echo "⚠️  IMPORTANT: Acceptez le certificat auto-signé dans votre navigateur"
echo "✋ Appuyez sur CTRL+C pour arrêter"

# Démarrer l'application
node server/app.js
```

- Script de test complet (**scripts/test-app.js**)

nano scripts/test-app.js

```
root@machine1:/home/sergio/webrtc-complete-app# nano scripts/test-app.js
root@machine1:/home/sergio/webrtc-complete-app#
```

```
#!/usr/bin/env node

const https = require('https');
const fs = require('fs');

console.log('🧪 Test de l\'Application WebRTC Complète');

// Configuration du test
const baseUrl = 'https://localhost:8443';
const endpoints = [
    '/',
    '/api/health',
    '/api/stats',
    '/api/rooms',
    '/api/test'
];

// Ignorer les certificats auto-signés pour les tests
process.env["NODE_TLS_REJECT_UNAUTHORIZED"] = 0;

async function testEndpoint(endpoint) {
    return new Promise((resolve) => {
        const url = `${baseUrl}${endpoint}`;

        https.get(url, (res) => {
            let data = '';
            res.on('data', chunk => data += chunk);
            res.on('end', () => {
                resolve({
                    endpoint,
                    status: res.statusCode,
                    success: res.statusCode === 200,
                    data: data
                });
            });
        }).on('error', (error) => {
            resolve({
                endpoint,
                status: 0,
                success: false,
                error: error.message
            });
        });
    });
}

async function runTests() {
    console.log('\n📡 Test des endpoints...\n');
```

```
    for (const endpoint of endpoints) {
        const result = await testEndpoint(endpoint);

        if (result.success) {
            console.log(`✅ ${endpoint} - HTTP ${result.status}`);
        } else {
            console.log(`❌ ${endpoint} - ${result.error || 'HTTP ' +
result.status}`);
        }
    }

    console.log('\n📊 Test des fonctionnalités...');

    // Test de création de salle via API
    try {
        const postData = JSON.stringify({
            roomId: 'test-room',
            maxClients: 5,
            metadata: { name: 'Salle de Test' }
        });

        const options = {
            hostname: 'localhost',
            port: 8443,
            path: '/api/rooms',
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Content-Length': postData.length
            },
            rejectUnauthorized: false
        };

        const req = https.request(options, (res) => {
            if (res.statusCode === 201) {
                console.log('✅ Création de salle - OK');
            } else {
                console.log('❌ Création de salle - ÉCHEC');
            }
        });

        req.on('error', () => {
            console.log('❌ Création de salle - ERREUR');
        });

        req.write(postData);
        req.end();

    } catch (error) {
```

```
        console.log('❌ Création de salle - ERREUR:', error.message);
    }

    console.log('\n🚀 Tests terminés !');
    console.log('🔗 Ouvrez https://localhost:8443 dans votre navigateur');
    console.log('🔐 Acceptez le certificat auto-signé');
}

// Attendre 2 secondes pour que le serveur démarre
setTimeout(runTests, 2000);
```

6. Commandes de Déploiement

- Installation complète

Créons les scripts exécutables par la commande suivante

chmod +x scripts/*.sh

```
root@machine1:/home/sergio/webrtc-complete-app# chmod +x scripts/*.sh
root@machine1:/home/sergio/webrtc-complete-app#
```

- Démarrage de l'application

./scripts/start-app.sh

```
root@machine1:/home/sergio/webrtc-complete-app# ./scripts/start-app.sh
=== Démarrage de l'Application WebRTC Complète ===
🚀 Démarrage de l'application...
🔒 L'application sera accessible sur https://localhost:8443
⚠️IMPORTANT: Acceptez le certificat auto-signé dans votre navigateur
🖐️ Appuyez sur CTRL+C pour arrêter
🚀 Démarrage de l'application WebRTC complète...
✅ Certificats SSL chargés

🎉 Application WebRTC complète démarrée !
🔒 URL HTTPS: https://localhost:8443/
📡 Socket.IO: wss://localhost:8443/socket.io/
🔗 PeerJS: https://localhost:8443/peerjs
📊 API: https://localhost:8443/api/

🔐 IMPORTANT: Acceptez le certificat auto-signé dans votre navigateur
🖐️ Appuyez sur CTRL+C pour arrêter
```

7. Test de l'application

Ne pas arretez le scripte ci-dessus mais plutôt ouvrir un autre terminal pour le test

node scripts/test-app.js

```
root@machine1:/home/sergio/webrtc-complete-app# node scripts/test-app.js
🧪 Test de l'Application WebRTC Complète

📡 Test des endpoints...

(node:7451) Warning: Setting the NODE_TLS_REJECT_UNAUTHORIZED environment variable to '0' makes TLS connections and HTTPS
requests insecure by disabling certificate verification.
(Use `node --trace-warnings ...` to show where the warning was created)
✅ / - HTTP 200
✅ /api/health - HTTP 200
✅ /api/stats - HTTP 200
✅ /api/rooms - HTTP 200
✅ /api/test - HTTP 200

📊 Test des fonctionnalités...

🎉 Tests terminés !
🔗 Ouvrez https://localhost:8443 dans votre navigateur
🔐 Acceptez le certificat auto-signé
✅ Création de salle - OK
root@machine1:/home/sergio/webrtc-complete-app#
```

```
🔐 IMPORTANT: Acceptez le certificat auto-signé dans votre navigateur
✋ Appuyez sur CTRL+C pour arrêter

[ROOM] Salle créée: test-room (Salle de Test)
```

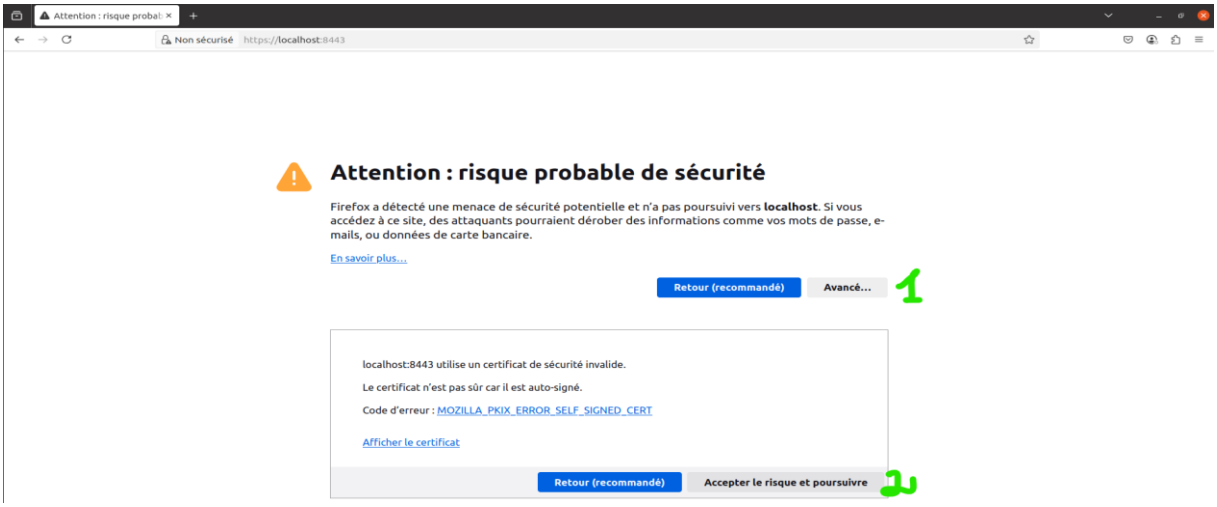
ufw allow 8443

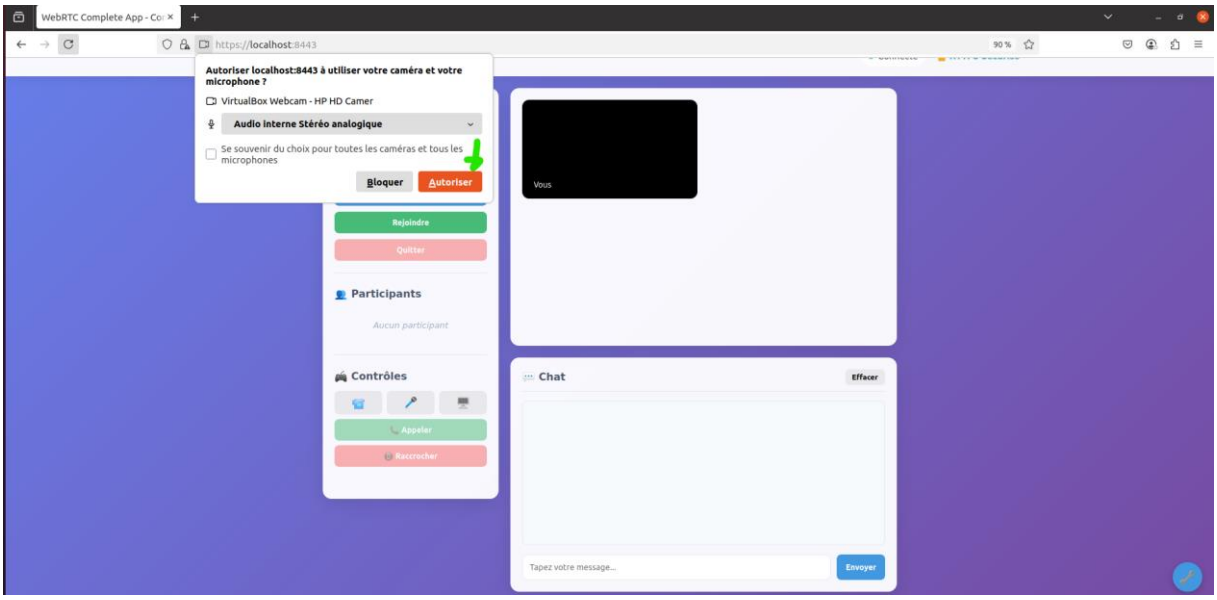
```
root@machine1:/home/sergio/webrtc-complete-app# ufw allow 8443
La règle a été ajoutée
La règle a été ajoutée (v6)
root@machine1:/home/sergio/webrtc-complete-app#
```

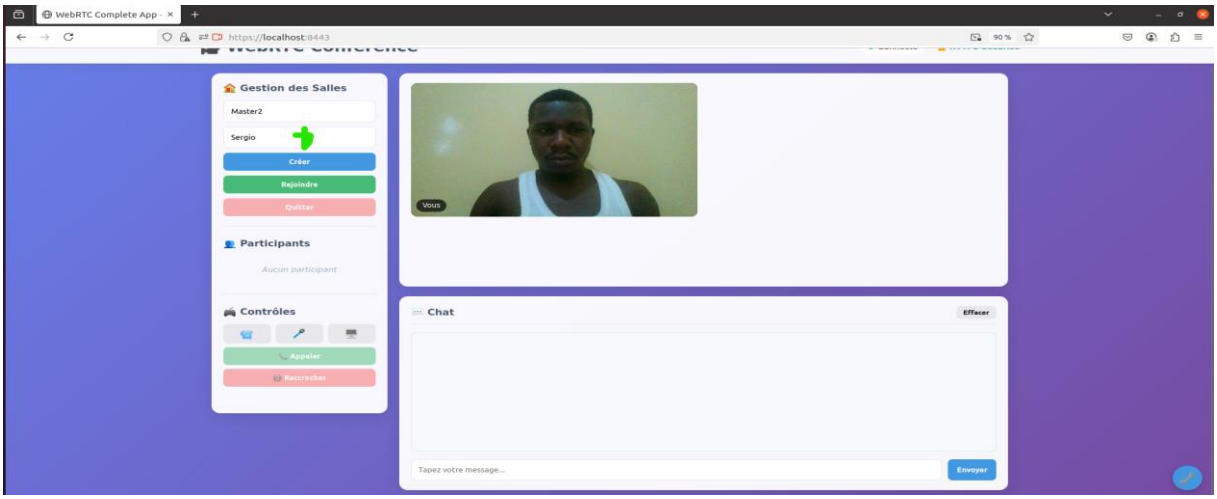
node server/app.js

```
root@machine1:/home/sergio/webrtc-complete-app# node server/app.js
🚀 Démarrage de l'application WebRTC complète...
✅ Certificats SSL chargés

🎉 Application WebRTC complète démarrée !
🔒 URL HTTPS: https://localhost:8443/
🔌 Socket.IO: wss://localhost:8443/socket.io/
🔗 PeerJS: https://localhost:8443/peerjs
📊 API: https://localhost:8443/api/

🔐 IMPORTANT: Acceptez le certificat auto-signé dans votre navigateur
✋ Appuyez sur CTRL+C pour arrêter
```

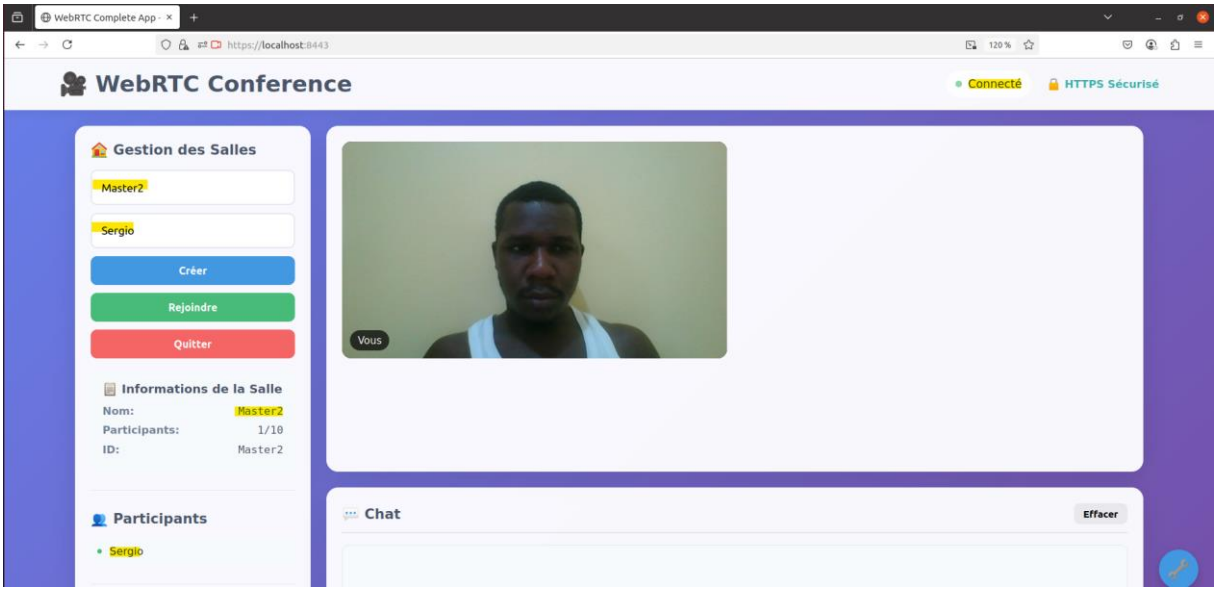
Ouvrons **https://localhost:8443** dans notre navigateur

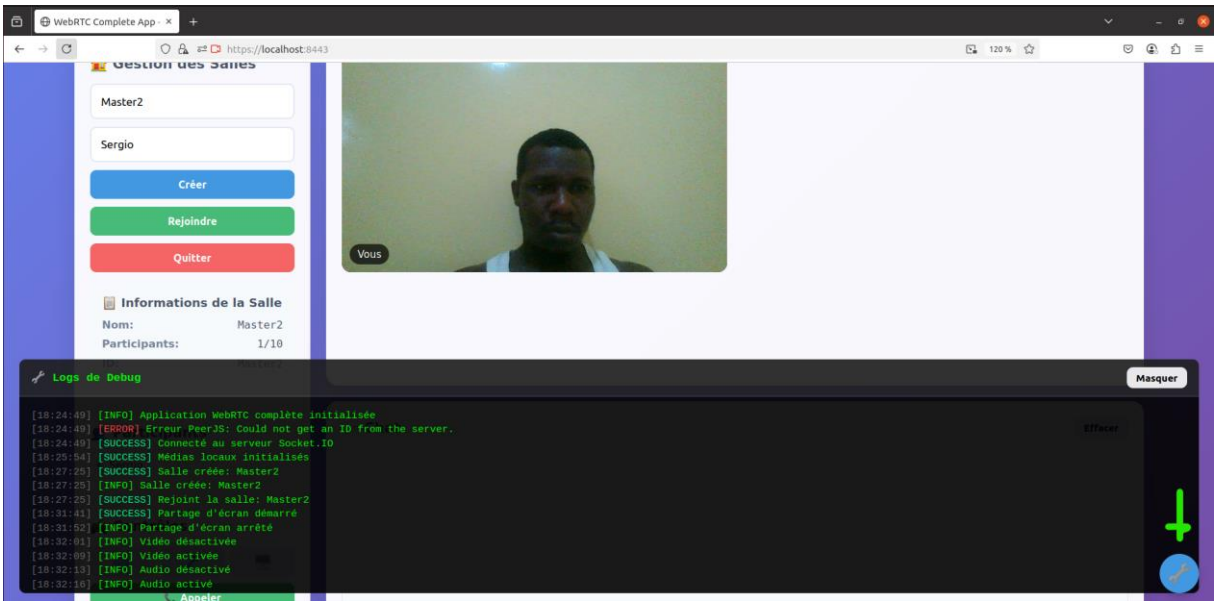




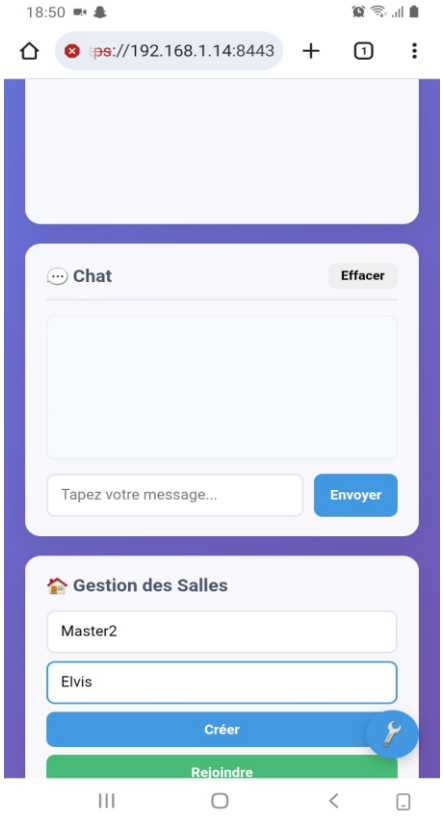


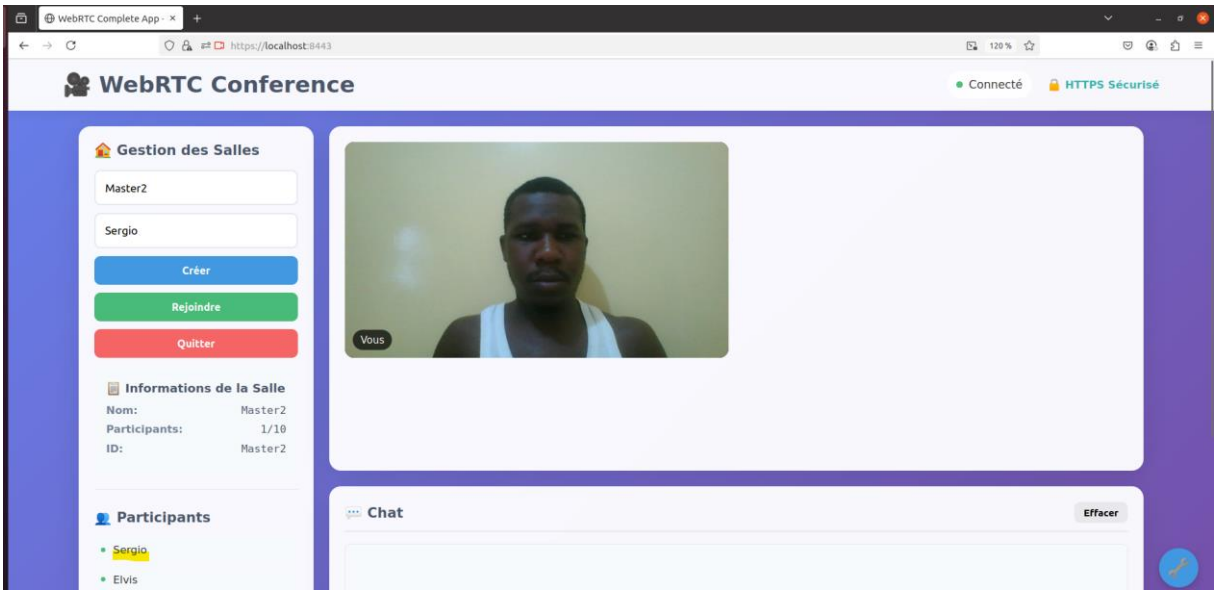

```
[SOCKET] Client connecté: p5E8k8TPaa6PXFwdAAAB
[ROOM] Salle créée: Master2 (Master2)
[ROOM] Client p5E8k8TPaa6PXFwdAAAB rejoint la salle Master2 (1/10)
```



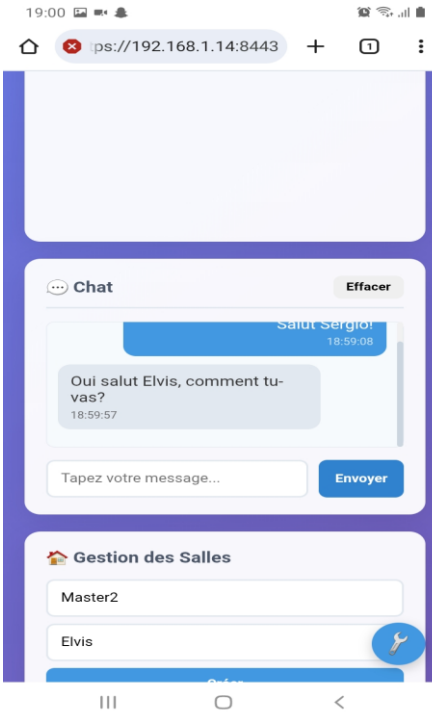

WebRTC Complete App
https://localhost:8443
Master2
Sergio
Créer
Rejoindre
Quitter
Informations de la Salle
Nom: Master2
Participants: 1/10
Vous
Logs de Debug
Masquer
[18:24:49] [INFO] Application WebRTC complète initialisée
[18:24:49] [ERROR] Erreur PeerJS: Could not get an ID from the server.
[18:24:49] [SUCCESS] Connecté au serveur Socket.IO
[18:25:54] [SUCCESS] Médias locaux initialisés
[18:27:25] [SUCCESS] Salle créée: Master2
[18:27:25] [INFO] Salle créée: Master2
[18:27:25] [SUCCESS] Rejoint la salle: Master2
[18:31:41] [SUCCESS] Partage d'écran démarré
[18:31:52] [INFO] Partage d'écran arrêté
[18:32:01] [INFO] Vidéo désactivée
[18:32:09] [INFO] Vidéo activée
[18:32:13] [INFO] Audio désactivé
[18:32:16] [INFO] Audio activé
Appeler

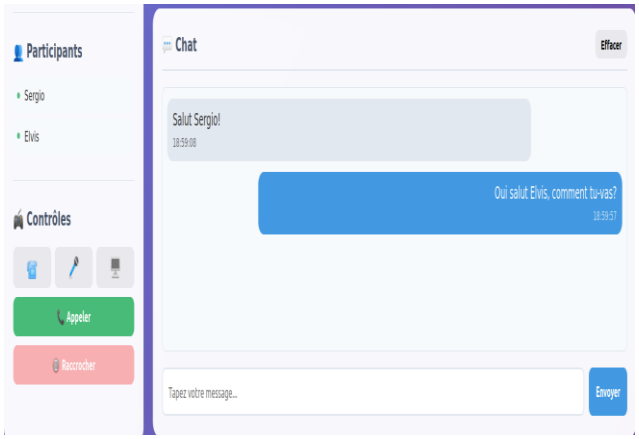

ps://192.168.1.14:8443
WebRTC Conference
Connecté
HTTPS Sécurisé
192.168.1.14:8443 souhaite utiliser votre appareil photo et votre micro
Autoriser pendant la visite du site
Autoriser cette fois-ci
Ne jamais autoriser
Chat
Effacer


18:50
ps://192.168.1.14:8443
Chat
Effacer
Tapez votre message...
Envoyer
Gestion des Salles
Master2
Elvis
Créer
Rejoindre



Le chat






```
[SOCKET] Client déconnecté: 8og2zxDkuGtdlbipAAAB
[SOCKET] Client connecté: nsDKKwbMRWZM0ILAAAAF
[ROOM] Salle créée: Master2 (Master2)
[ROOM] Client s49CTn80TQAAlr2-AAAD rejoint la salle Master2 (1/10)
[ROOM] Client nsDKKwbMRWZM0ILAAAAF rejoint la salle Master2 (2/10)
[SOCKET] Client déconnecté: nsDKKwbMRWZM0ILAAAAF
```

4. Création d'une Application Electron avec Gestion de Compte Président

4.1. Introduction à Electron

Electron est un framework open-source permettant de créer des applications de bureau multiplateformes avec des technologies web (HTML, CSS, JavaScript). Il utilise Node.js pour le backend et Chromium pour le rendu de l'interface utilisateur. Dans ce projet, nous allons :

- Créer une interface utilisateur pour gérer un compte président.
- Intégrer une base de données SQLite pour stocker les données.
- Implémenter des fonctionnalités CRUD (Create, Read, Update, Delete).
- Générer des exécutables pour Windows et Linux.

4.2. Prérequis

- **Node.js** (version 18 ou supérieure) et npm installés.
- Un éditeur de code (par exemple, VS Code).
- **SQLite** pour la gestion de la base de données.
- **Electron Builder** pour la génération des exécutables.

4.3. Configuration Initiale du Projet

4.3.1 Création de répertoire du projet

On créer un dossier pour notre projet et initialisez-le avec npm :

mkdir electron-president-app

cd electron-president-app/

```
root@machine2:/home/oracle# mkdir electron-president-app
root@machine2:/home/oracle# cd electron-president-app/
root@machine2:/home/oracle/electron-president-app#
```

npm init

```
root@machine2:/home/oracle/electron-president-app# npm init -y
Wrote to /home/oracle/electron-president-app/package.json:

{
  "name": "electron-president-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

4.3.2 Installation des dépendances

Installez Electron, SQLite3, et Electron Builder :

npm install electron sqlite3 electron-builder --save-dev

```
root@machine2:/home/oracle/electron-president-app# npm install electron sqlite3 electron-builder --save-dev
npm warn skipping integrity check for git dependency ssh://git@github.com/electron/node-gyp.git
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way
 to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@8.1.0: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@8.1.0: Glob versions prior to v9 are no longer supported
npm warn deprecated are-we-there-yet@3.0.1: This package is no longer supported.
npm warn deprecated npmlog@6.0.2: This package is no longer supported.
npm warn deprecated @npmcli/move-file@2.0.1: This functionality has been moved to @npmcli/fs
npm warn deprecated gauge@4.0.4: This package is no longer supported.
npm warn deprecated boolean@3.2.0: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs
npm warn deprecated rimraf@2.6.3: Rimraf versions prior to v4 are no longer supported

added 421 packages, and audited 422 packages in 3m

65 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Modifiez le fichier **package.json** pour inclure les scripts nécessaires

nano package.json

```
root@machine2:/home/oracle/electron-president-app# nano package.json
root@machine2:/home/oracle/electron-president-app#
```

```
{
  "name": "electron-president-app",
  "version": "1.0.0",
  "description": "Application de gestion de compte président avec Electron",
  "main": "main.js",
  "scripts": {
    "start": "electron . -no-sandbox",
    "build": "electron-builder --win --linux"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "electron": "^26.2.0",
    "electron-builder": "^24.6.4",
    "sqlite3": "^5.1.6"
  },
  "build": {
    "appId": "com.example.presidentapp",
    "productName": "President App",
    "win": {
      "target": ["nsis"]
    },
    "linux": {
      "target": ["AppImage", "deb"]
    }
  }
}
```

npm install

```
root@machine2:/home/oracle/electron-president-app# npm install

added 34 packages, removed 76 packages, changed 24 packages, and audited 380 packages in 2m

52 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

4.4 Création de l'Interface Utilisateur

- Création de fichier principal **main.js**

Ce fichier configure la fenêtre principale de l'application Electron.

nano main.js

```
root@machine2:/home/oracle/electron-president-app# nano main.js
root@machine2:/home/oracle/electron-president-app#
```

```
const { app, BrowserWindow } = require('electron');
const path = require('path');

// Fonction pour créer la fenêtre principale
function createWindow() {
    // Création d'une nouvelle fenêtre avec des dimensions et préférences
définies
    const win = new BrowserWindow({
        width: 800,
        height: 600,
        webPreferences: {
            nodeIntegration: true,      // Active l'intégration de Node.js
            contextIsolation: false     // Désactive l'isolation du contexte
pour un accès direct à Node.js
        }
    });

    // Charge le fichier HTML principal
    win.loadFile('index.html');
}

// Événement déclenché lorsque Electron est prêt
app.whenReady().then(() => {
    createWindow();

    // Gestion de l'événement 'activate' (principalement pour macOS)
    app.on('activate', () => {
        // Recrée une fenêtre si aucune n'est ouverte
        if (BrowserWindow.getAllWindows().length === 0) {
```

```
            createWindow();
        }
    });
});

// Ferme l'application lorsque toutes les fenêtres sont fermées (sauf sur
macOS)
app.on('window-all-closed', () => {
    if (process.platform !== 'darwin') {
        app.quit();
    }
});
```

Créez un fichier **index.html** pour l'interface utilisateur avec un formulaire CRUD.

- **Gérer la logique frontend renderer.js**

Créez un fichier renderer.js pour gérer les interactions avec la base de données SQLite

nano index.html

```
root@machine2:/home/oracle/electron-president-app# nano index.html
root@machine2:/home/oracle/electron-president-app#
```

```
<!DOCTYPE html>
<html>
<head>
  <title>President App</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      background-color: #f4f4f9;
    }
    h1 {
      color: #333;
      text-align: center;
    }
    .form-group {
      margin-bottom: 20px;
      text-align: center;
    }
    input {
      padding: 10px;
      margin: 5px;
      width: 200px;
      border: 1px solid #ccc;
      border-radius: 4px;
```

```
    }
    button {
      padding: 10px 15px;
      margin: 5px;
      background-color: #007bff;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background-color: #0056b3;
    }
    table {
      width: 80%;
      margin: 20px auto;
      border-collapse: collapse;
      background-color: white;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    th, td {
      border: 1px solid #ddd;
      padding: 12px;
      text-align: left;
    }
    th {
      background-color: #007bff;
      color: white;
    }
    tr:nth-child(even) {
      background-color: #f9f9f9;
    }
    .action-buttons button {
      padding: 8px;
      margin: 2px;
      background-color: #28a745;
    }
    .action-buttons button.delete {
      background-color: #dc3545;
    }
    .action-buttons button:hover {
      opacity: 0.9;
    }
  </style>
</head>
<body>
  <h1>Gestion du Compte Président</h1>
  <div class="form-group">
    <input type="text" id="name" placeholder="Nom du président" />
```

```
    <input type="text" id="email" placeholder="Email" />
    <button onclick="addPresident()">Ajouter Président</button>
  </div>
  <table id="presidentTable">
    <thead>
      <tr>
        <th>ID</th>
        <th>Nom</th>
        <th>Email</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody id="presidentList"></tbody>
  </table>

  <script src="renderer.js"></script>
</body>
</html>
```

nano renderer.js

```
root@machine2:/home/oracle/electron-president-app# nano renderer.js
root@machine2:/home/oracle/electron-president-app#
```

```
const sqlite3 = require('sqlite3').verbose();
const db = new sqlite3.Database('./presidents.db');

db.serialize(() => {
  db.run(`CREATE TABLE IF NOT EXISTS presidents (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    email TEXT NOT NULL
  )`);
});

function addPresident() {
  const name = document.getElementById('name').value.trim();
  const email = document.getElementById('email').value.trim();
  if (name && email) {
    db.run(`INSERT INTO presidents (name, email) VALUES (?, ?)`, [name,
email], function(err) {
      if (err) {
        console.error('Erreur lors de l\'ajout:', err.message);
        alert('Erreur lors de l\'ajout du président.');
      } else {
        document.getElementById('name').value = '';
        document.getElementById('email').value = '';
        loadPresidents();
      }
```

```
    });
  } else {
    alert('Veuillez remplir tous les champs.');
  }
}

function loadPresidents() {
  const presidentList = document.getElementById('presidentList');
  presidentList.innerHTML = '';
  db.all(`SELECT * FROM presidents`, [], (err, rows) => {
    if (err) {
      console.error('Erreur lors du chargement:', err.message);
      alert('Erreur lors du chargement des données.');
    }
    rows.forEach(row => {
      const tr = document.createElement('tr');
      tr.innerHTML = `
        <td>${row.id}</td>
        <td>${row.name}</td>
        <td>${row.email}</td>
        <td class="action-buttons">
          <button onclick="editPresident(${row.id}, '${row.name}',
'${row.email}')">Modifier</button>
          <button class="delete"
onclick="deletePresident(${row.id})">Supprimer</button>
        </td>
      `;
      presidentList.appendChild(tr);
    });
  });
}

function editPresident(id, name, email) {
  const newName = prompt('Modifier le nom:', name);
  const newEmail = prompt('Modifier l\'email:', email);
  if (newName && newEmail) {
    db.run(`UPDATE presidents SET name = ?, email = ? WHERE id = ?`,
[newName.trim(), newEmail.trim(), id], function(err) {
      if (err) {
        console.error('Erreur lors de la modification:', err.message);
        alert('Erreur lors de la modification du président.');
      } else {
        loadPresidents();
      }
    });
  } else {
    alert('Veuillez fournir des valeurs valides.');
  }
}
```

```
function deletePresident(id) {
  if (confirm('Voulez-vous vraiment supprimer ce président ?')) {
    db.run(`DELETE FROM presidents WHERE id = ?`, [id], function(err) {
      if (err) {
        console.error('Erreur lors de la suppression:', err.message);
        alert('Erreur lors de la suppression du président.');
      } else {
        loadPresidents();
      }
    });
  }
}

document.addEventListener('DOMContentLoaded', loadPresidents);
```

sudo chown -R oracle:oracle /home/oracle/electron-president-app

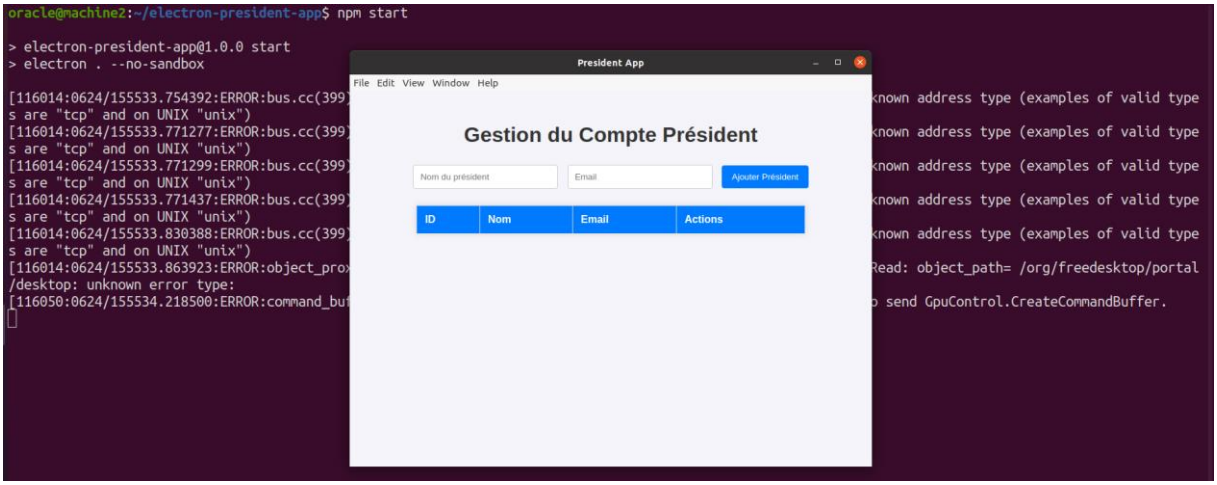
chmod -R 755 /home/oracle/electron-president-app

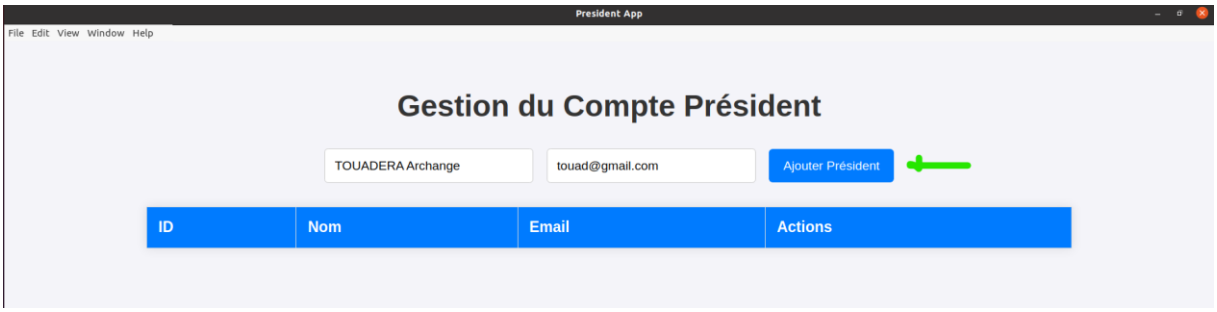
```
root@machine2:/home/oracle/electron-president-app# sudo chown -R oracle:oracle /home/oracle/electron-president-app
root@machine2:/home/oracle/electron-president-app# chmod -R 755 /home/oracle/electron-president-app
root@machine2:/home/oracle/electron-president-app#
```

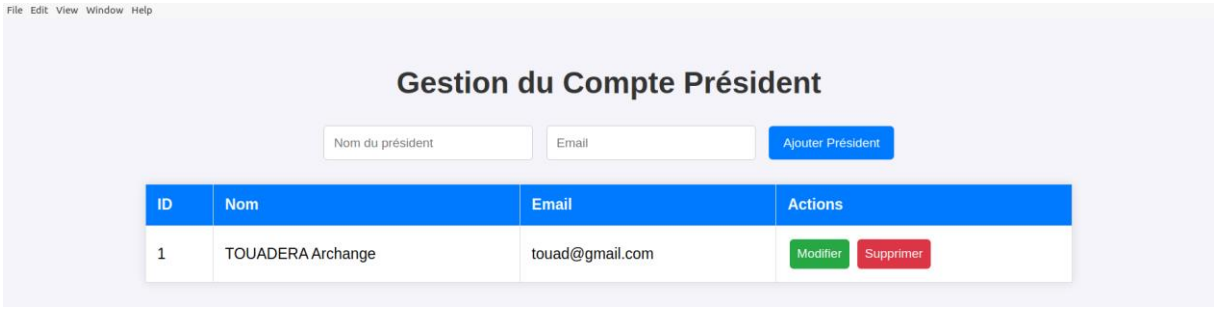
- Lancer l'application

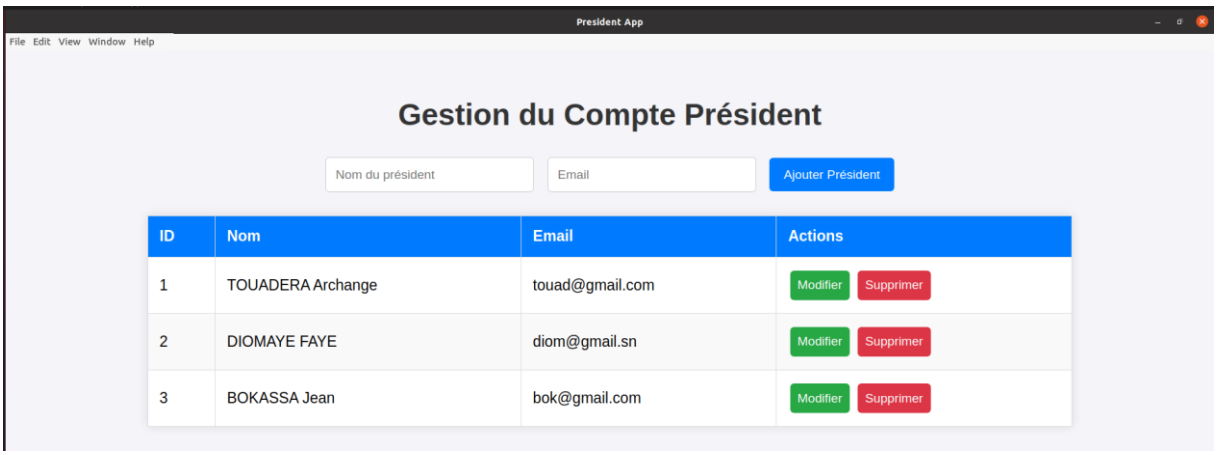
$ npm start

```
oracle@machine2:~/electron-president-app$ npm start

> electron-president-app@1.0.0 start
> electron . --no-sandbox
```

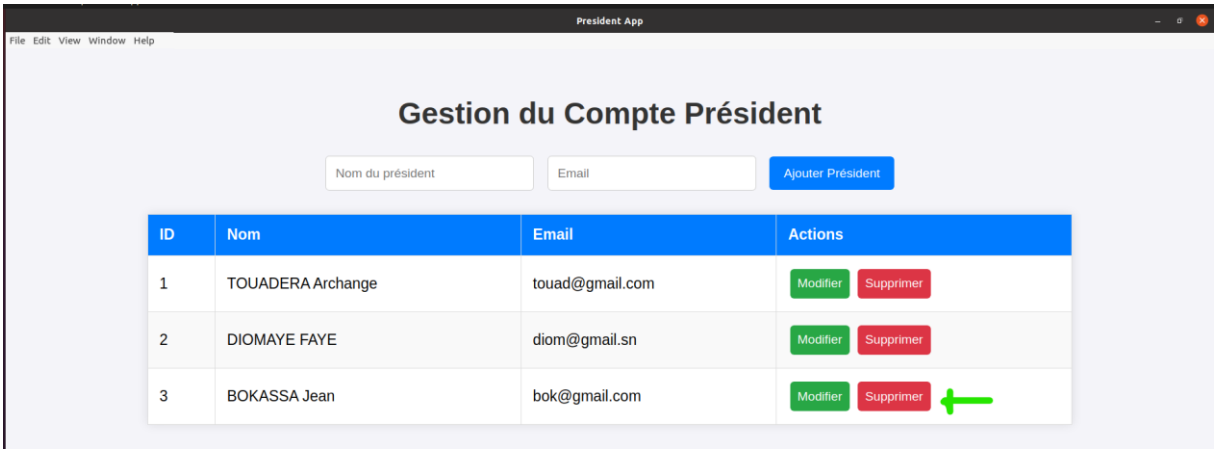


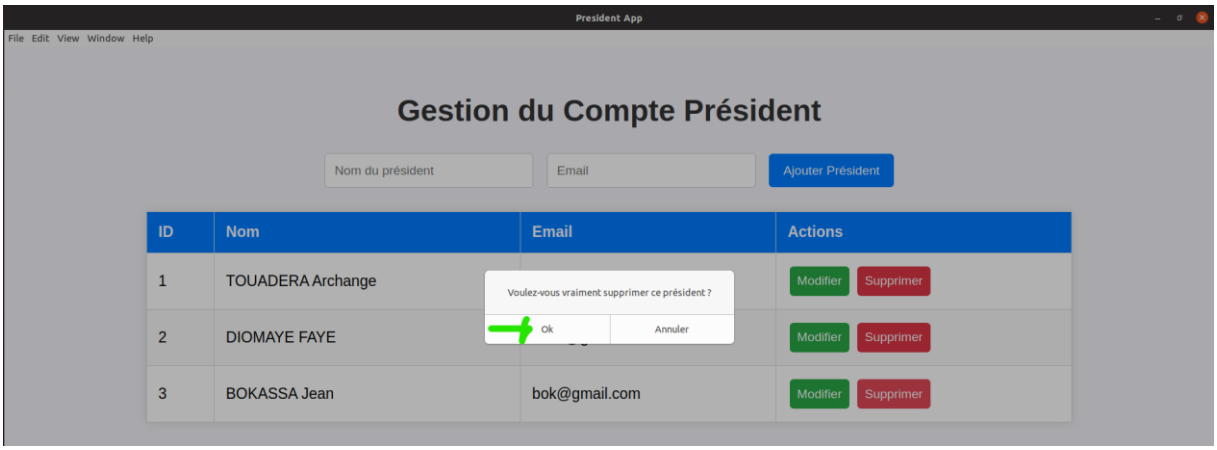


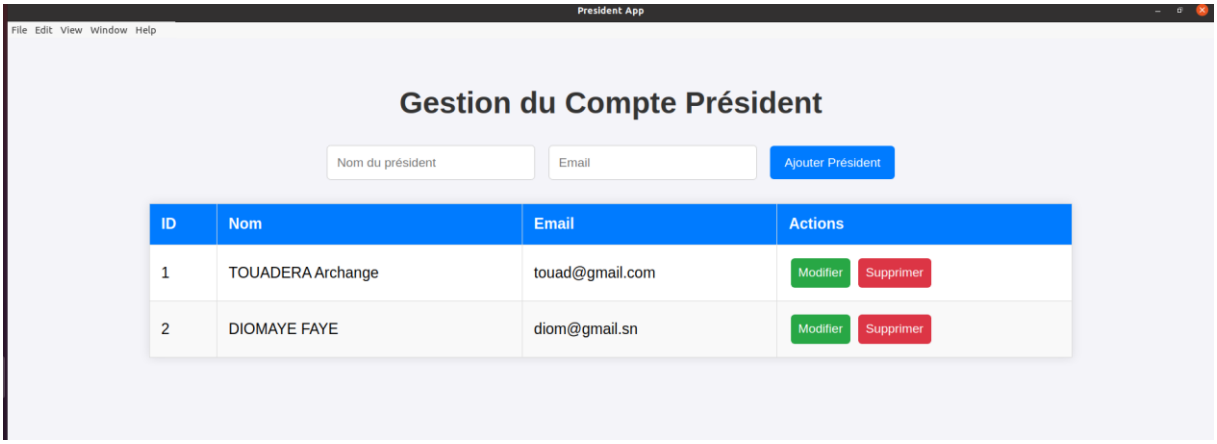




- Suppression







5. Génération des Exécutables pour Windows et Linux

5.1 Configurer Electron Builder

Le fichier package.json inclut déjà la configuration pour Electron Builder. Les cibles sont :

- **Windows** : NSIS (installateur).
- **Linux** : AppImage et deb.

5.2 Générer les exécutables

Exécutez la commande suivante pour générer les exécutables pour Windows et Linux

Installation de Wine

sudo apt-get install -y wine winetricks

```
root@machine2:/home/oracle/electron-president-app# sudo apt-get install -y wine winetricks
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
wine est déjà la version la plus récente (5.0-3ubuntu1).
Les paquets supplémentaires suivants seront installés :
  cabextract fuseiso libmspack0 p7zip p7zip-full
```

wine --version

```
root@machine2:/home/oracle/electron-president-app# wine --version
wine-5.0 (Ubuntu 5.0-3ubuntu1)
root@machine2:/home/oracle/electron-president-app#
```

sudo apt-get install -y libarchive-tools fakeroot

```
root@machine2:/home/oracle/electron-president-app# sudo apt-get install -y libarchive-tools fakeroot
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
fakeroot est déjà la version la plus récente (1.24-1).
fakeroot passé en « installé manuellement ».
Les NOUVEAUX paquets suivants seront installés :
  libarchive-tools
```

Corrigé de package.json

- Homepage manquante : Ajout du champ homepage
- Email de l'auteur manquant : Transformation du champ author en objet avec name et email
- Mainteneur Linux manquant : Ajout du champ maintainer dans la section linux
- Catégorie Linux : Ajout d'une catégorie pour éviter l'avertissement

Changements à personnaliser

Remplacez les valeurs suivantes par vos vraies informations :

- "Votre Nom" → votre vrai nom
- "votre.email@example.com" → votre vraie adresse email
- "https://github.com/votre-username/electron-president-app" → l'URL de votre projet (ou une URL de votre choix)

nano package.json

```
root@machine2:/home/oracle/electron-president-app# nano package.json
root@machine2:/home/oracle/electron-president-app#
```

```
{
  "name": "electron-president-app",
  "version": "1.0.0",
  "description": "Application de gestion de compte président avec Electron",
  "main": "main.js",
  "homepage": "https://github.com/votre-username/electron-president-app",
  "scripts": {
    "start": "electron . --no-sandbox",
    "build": "electron-builder --win --linux"
```

```
  },
  "author": {
    "name": "Sergio",
    "email": "sergio@gmail.sn"
  },
  "license": "ISC",
  "devDependencies": {
    "electron": "^26.2.0",
    "electron-builder": "^24.6.4",
    "sqlite3": "^5.1.6"
  },
  "build": {
    "appId": "com.example.presidentapp",
    "productName": "President App",
    "win": {
      "target": ["nsis"]
    },
    "linux": {
      "target": ["AppImage", "deb"],
      "maintainer": "Sergio <ssergio@gmail.sn>",
      "category": "Office"
    }
  }
}
```

npm run build

```
root@machine2:/home/oracle/electron-president-app# npm run build

> electron-president-app@1.0.0 build
> electron-builder --win --linux

  • electron-builder  version=24.13.3 os=5.15.0-139-generic
  • loaded configuration  file=package.json ("build" field)
  • writing effective config  file=dist/builder-effective-config.yaml
  • packaging       platform=linux arch=x64 electron=26.6.10 appOutDir=dist/linux-unpacked
  • building        target=AppImage arch=x64 file=dist/President App-1.0.0.AppImage
  • default Electron icon is used  reason=application icon is not set
  • packaging       platform=win32 arch=x64 electron=26.6.10 appOutDir=dist/win-unpacked
  • default Electron icon is used  reason=application icon is not set
  • building        target=deb arch=x64 file=dist/electron-president-app_1.0.0_amd64.deb
  • downloading     url=https://github.com/electron-userland/electron-builder-binaries/releases/download/fpm-1.9.3-2.3.1-linux-x86_64/fpm-1.9.3-2.3.1-li
nux-x86_64.7z size=5.0 MB parts=1
  • downloaded      url=https://github.com/electron-userland/electron-builder-binaries/releases/download/fpm-1.9.3-2.3.1-linux-x86_64/fpm-1.9.3-2.3.1-li
nux-x86_64.7z duration=4.952s
  • building        target=nsis file=dist/President App Setup 1.0.0.exe archs=x64 oneClick=true perMachine=false
  • downloading     url=https://github.com/electron-userland/electron-builder-binaries/releases/download/nsis-3.0.4.1/nsis-3.0.4.1.7z size=1.3 MB parts=
1
  • downloaded      url=https://github.com/electron-userland/electron-builder-binaries/releases/download/nsis-3.0.4.1/nsis-3.0.4.1.7z duration=2.686s
  • downloading     url=https://github.com/electron-userland/electron-builder-binaries/releases/download/nsis-resources-3.4.1/nsis-resources-3.4.1.7z si
ze=731 kB parts=1
  • downloaded      url=https://github.com/electron-userland/electron-builder-binaries/releases/download/nsis-resources-3.4.1/nsis-resources-3.4.1.7z du
ration=2.704s
  • building block map  blockMapFile=dist/President App Setup 1.0.0.exe.blockmap
root@machine2:/home/oracle/electron-president-app#
```

Parfait ! Notre build fonctionne maintenant correctement ! 🎉

Ce qui se passe dans cette sortie :

✅ Succès - Plus d'erreurs ! Votre package.json corrigé a résolu tous les problèmes précédents.

Processus en cours :

Linux :

✓ AppImage créé : President App-1.0.0.AppImage

✓ Package Debian créé : electron-president-app_1.0.0_amd64.deb

Windows :

✓ Installateur NSIS créé : President App Setup 1.0.0.exe

🔄 Création du block map en cours (presque terminé)

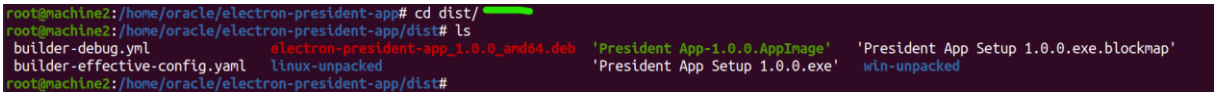
Avertissements (non bloquants) :

Icône par défaut utilisée : Vous pouvez ajouter une icône personnalisée plus tard si vous le souhaitez

Fichiers de distribution générés :

- **Pour Linux** :
- **President App-1.0.0.AppImage** - Version portable (double-cliquez pour lancer)
- **electron-president-app_1.0.0_amd64.**deb - Package pour Ubuntu/Debian (installez avec dpkg -i)
- **Pour Windows** :
- **President App Setup 1.0.0.exe** - Installateur Windows complet
- **President App Setup 1.0.0.exe.blockmap** - Fichier de métadonnées (pour les mises à jour)
- **Versions de développement** :
- **linux-unpacked/** - Version Linux non-packagée (pour debug)
- **win-unpacked/** - Version Windows non-packagée (pour debug)

cd dist/



Comment tester notre application

Sur notre machine Linux actuelle

Rendez l'AppImage exécutable

chmod +x "President App-1.0.0.AppImage"

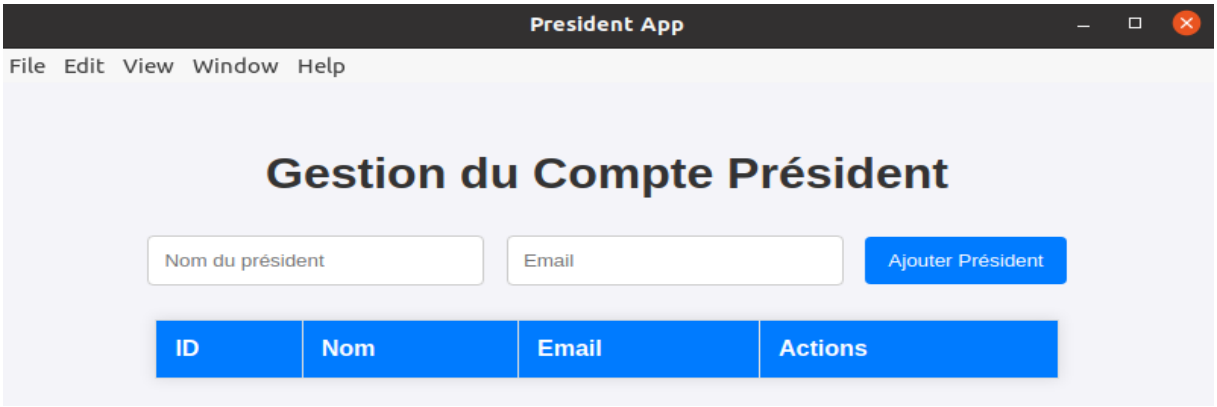
```
root@machine2:/home/oracle/electron-president-app/dist# chmod +x "President App-1.0.0.AppImage"
root@machine2:/home/oracle/electron-president-app/dist#
```

On lance l'application comme suite

$ cd dist/

$ ./"President App-1.0.0.AppImage"

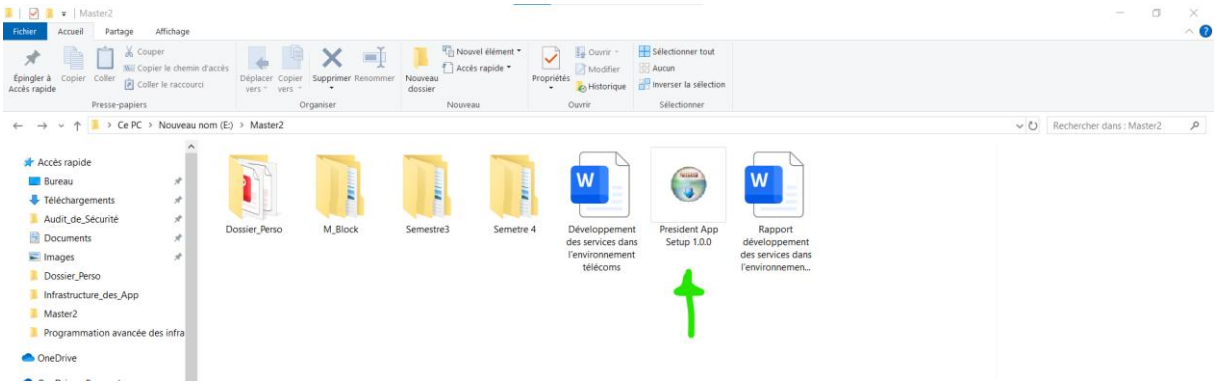
```
oracle@machine2:~/electron-president-app$ cd dist/
oracle@machine2:~/electron-president-app/dist$ ./"President App-1.0.0.AppImage"
[139706:0624/180405.873321:ERROR:bus.cc(399)] Failed to connect to the bus: Could not parse server address: Unknown address type (examples of valid type
s are "tcp" and on UNIX "unix")
[139706:0624/180405.873496:ERROR:bus.cc(399)] Failed to connect to the bus: Could not parse server address: Unknown address type (examples of valid type
s are "tcp" and on UNIX "unix")
[139706:0624/180405.873519:ERROR:bus.cc(399)] Failed to connect to the bus: Could not parse server address: Unknown address type (examples of valid type
s are "tcp" and on UNIX "unix")
[139706:0624/180405.873532:ERROR:bus.cc(399)] Failed to connect to the bus: Could not parse server address: Unknown address type (examples of valid type
s are "tcp" and on UNIX "unix")
[139706:0624/180406.646048:ERROR:bus.cc(399)] Failed to connect to the bus: Could not parse server address: Unknown address type (examples of valid type
s are "tcp" and on UNIX "unix")
[139706:0624/180406.818544:ERROR:object_proxy.cc(590)] Failed to call method: org.freedesktop.portal.Settings.Read: object_path= /org/freedesktop/portal
/desktop: unknown error type:
libva error: vaGetDriverNameByIndex() failed with unknown libva error, driver_name = (null)
```

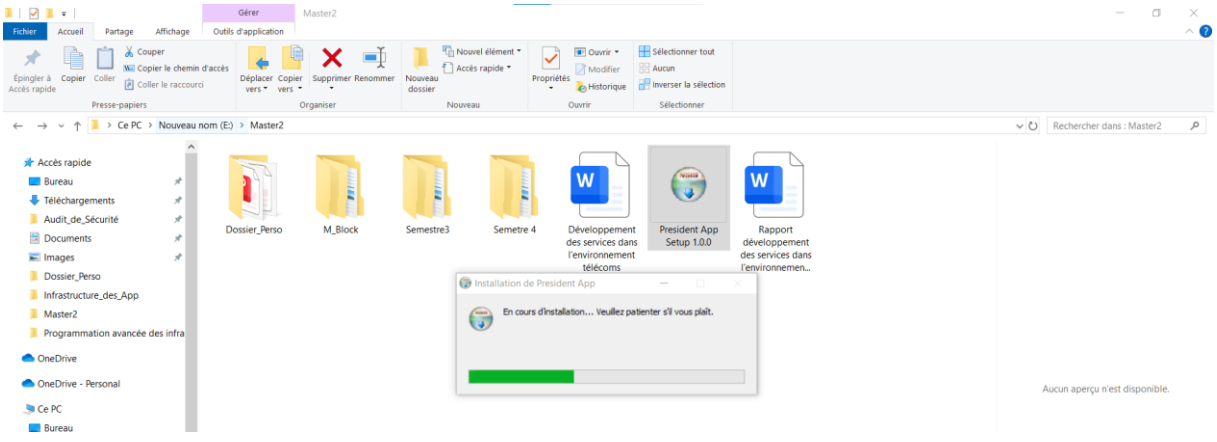


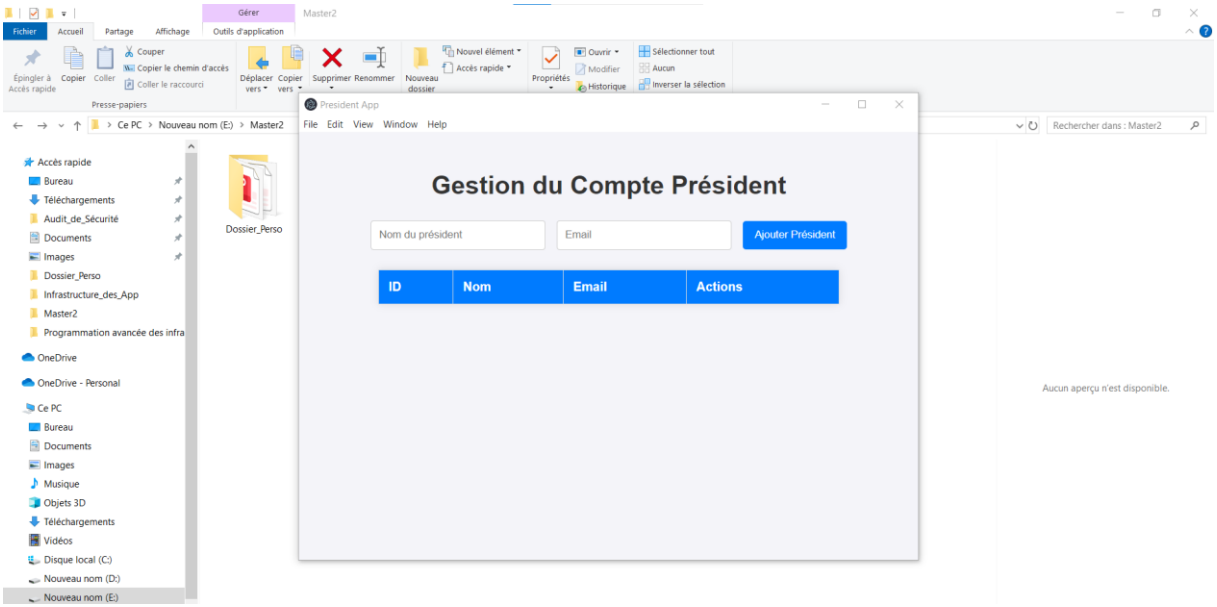
sudo dpkg -i electron-president-app_1.0.0_amd64.deb

```
root@machine2:/home/oracle/electron-president-app/dist# sudo dpkg -i electron-president-app_1.0.0_amd64.deb
Sélection du paquet electron-president-app précédemment désélectionné.
(Lecture de la base de données... 353187 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de electron-president-app_1.0.0_amd64.deb ...
Dépaquetage de electron-president-app (1.0.0) ...
Paramétrage de electron-president-app (1.0.0) ...
update-alternatives est /usr/bin/update-alternatives
update-alternatives: utilisation de « /opt/President App/electron-president-app » pour fournir « /usr/bin/electron-president-app » (electron-president-a
pp) en mode automatique
Traitement des actions différées (« triggers ») pour gnome-menus (3.36.0-1ubuntu1) ...
Traitement des actions différées (« triggers ») pour desktop-file-utils (0.24-1ubuntu3) ...
Traitement des actions différées (« triggers ») pour mime-support (3.64ubuntu1) ...
Traitement des actions différées (« triggers ») pour hicolor-icon-theme (0.17-2) ...
root@machine2:/home/oracle/electron-president-app/dist#
```

Transférez le fichier President App Setup 1.0.0.exe sur une machine Windows et double-cliquez dessus.


Master2
Fichier
Accueil
Partage
Affichage
Ce PC > Nouveau nom (E:) > Master2
Rechercher dans : Master2
Accès rapide
Bureau
Téléchargements
Audit_de_Sécurité
Documents
Images
Dossier_Perso
Infrastructure_des_App
Master2
Programmation avancée des infra
OneDrive
Dossier_Perso
M_Block
Semestre3
Semetre 4
Développement des services dans l'environnement télécoms
President App Setup 1.0.0
Rapport développement des services dans l'environnemen...


Gérer
Master2
Fichier
Accueil
Partage
Affichage
Outils d'application
Ce PC > Nouveau nom (E:) > Master2
Rechercher dans : Master2
Accès rapide
Bureau
Téléchargements
Audit_de_Sécurité
Documents
Images
Dossier_Perso
Infrastructure_des_App
Master2
Programmation avancée des infra
OneDrive
OneDrive - Personal
Ce PC
Bureau
Dossier_Perso
M_Block
Semestre3
Semetre 4
Développement des services dans l'environnement télécoms
President App Setup 1.0.0
Rapport développement des services dans l'environnemen...
Installation de President App
En cours d'installation... Veuillez patienter s'il vous plaît.
Aucun aperçu n'est disponible.


Gérer
Master2
Fichier
Accueil
Partage
Affichage
Outils d'application
Ce PC > Nouveau nom (E:) > Master2
Rechercher dans : Master2
Accès rapide
Bureau
Téléchargements
Audit_de_Sécurité
Documents
Images
Dossier_Perso
Infrastructure_des_App
Master2
Programmation avancée des infra
OneDrive
OneDrive - Personal
Ce PC
Bureau
Documents
Images
Musique
Objets 3D
Téléchargements
Vidéos
Disque local (C:)
Nouveau nom (D:)
Nouveau nom (E:)
Dossier_Perso
President App
File
Edit
View
Window
Help
Gestion du Compte Président
Nom du président
Email
Ajouter Président
ID
Nom
Email
Actions
Aucun aperçu n'est disponible.

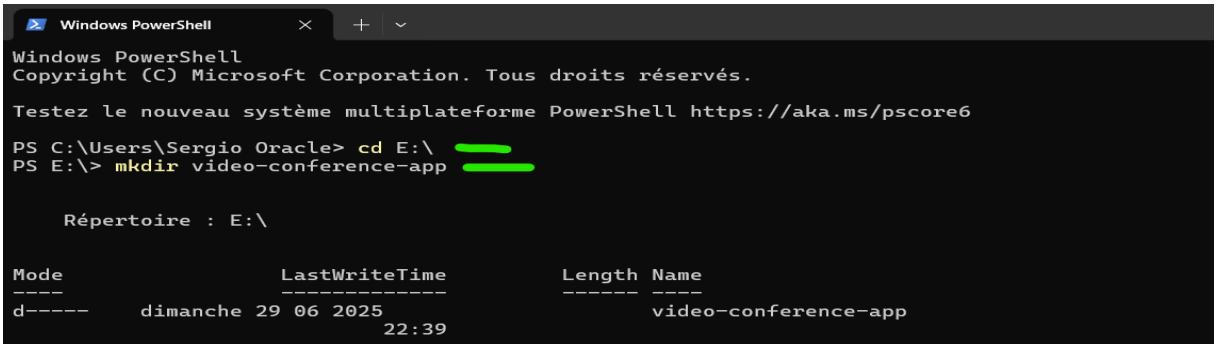
6. Application de Visioconférence Simple avec Electron et WebRTC

6.1 Prérequis

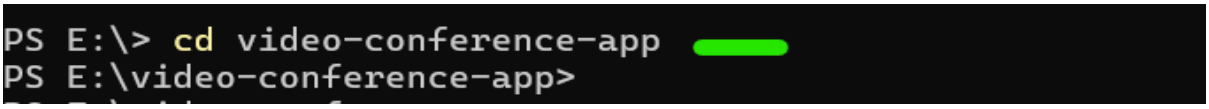
- **Node.js** (version 20.19 ou supérieure)
- **npm** (inclus avec Node.js)
- Un serveur de signalisation (nous utiliserons **Socket.IO** pour la simplicité)

> cd E:\

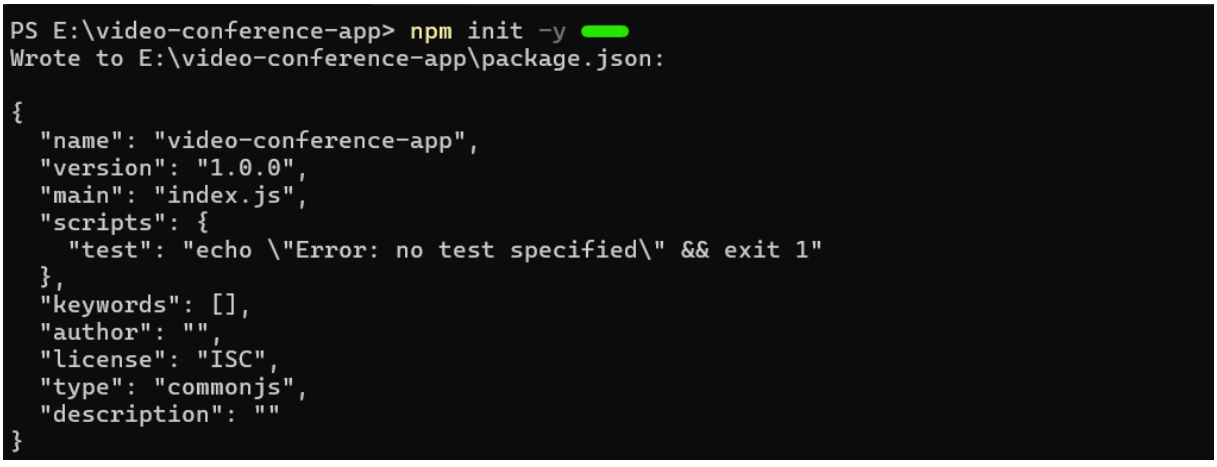
> mkdir video-conference-app



> cd video-conference-app

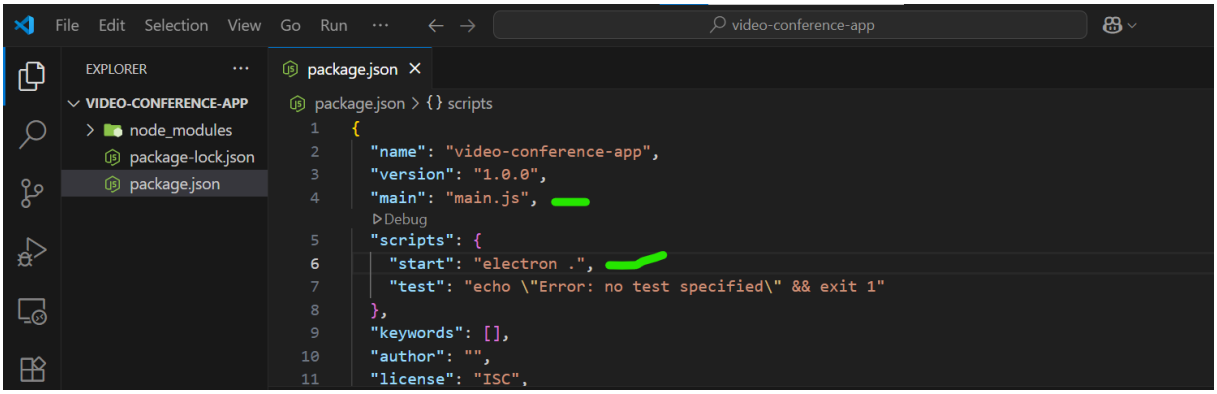


> npm init -y



> npm install electron socket.io-client webrtc-adapter

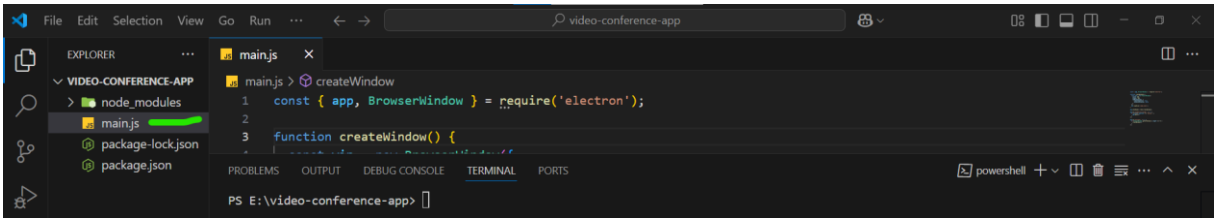
Le fichier **package.json**



> npm install

```
PS E:\video-conference-app> npm install

up to date, audited 83 packages in 5s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

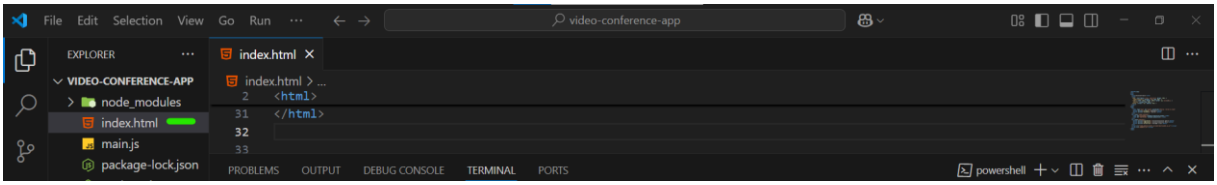
Fichier : **main.js**




```
const { app, BrowserWindow } = require('electron');

function createWindow() {
  const win = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false
    }
  });
  win.loadURL('https://localhost:3000', {
    extraHeaders: 'pragma: no-cache\n'
  });

  // Ignorer les erreurs de certificat pour le développement
  win.webContents.session.setCertificateVerifyProc((request, callback) => {
    callback(0); // Accepter tous les certificats
  });
```

```
  // Gestion des erreurs de chargement
  win.webContents.on('did-fail-load', (event, errorCode, errorDescription) =>
{
    console.error(`Erreur de chargement : ${errorDescription} (Code:
${errorCode})`);
  });
}

app.whenReady().then(createWindow);

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow();
  }
});
```

Fichier : **index.html**

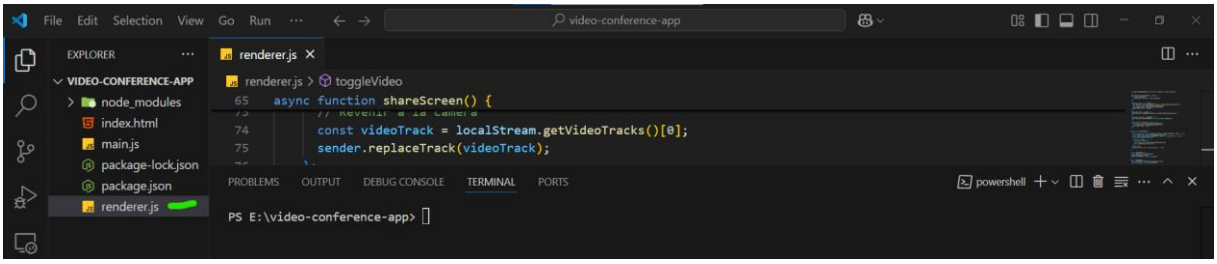


Le contenu du fichier

```
<!DOCTYPE html>
<html>
<head>
  <title>Visioconférence</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    .video-container { display: flex; gap: 20px; flex-wrap: wrap; }
    video { width: 300px; height: 200px; border: 1px solid #ccc; background:
black; }
    .controls { margin-top: 20px; }
    button { margin: 5px; padding: 10px; cursor: pointer; }
    input { padding: 8px; margin-right: 10px; }
  </style>
</head>
<body>
  <div>
    <input type="text" id="roomId" placeholder="Entrez l'ID de la room">
```

```
    <button onclick="joinRoom()">Rejoindre</button>
    <button onclick="leaveRoom()">Quitter</button>
  </div>
  <div class="video-container" id="videoContainer">
    <video id="localVideo" autoplay playsinline muted></video>
  </div>
  <div class="controls">
    <button onclick="toggleVideo()">Activer/Désactiver Caméra</button>
    <button onclick="toggleAudio()">Activer/Désactiver Micro</button>
    <button onclick="shareScreen()">Partager Écran</button>
  </div>
  <script src="/socket.io/socket.io.js"></script>
  <script src="renderer.js"></script>
</body>
</html>
```

Fichier : **renderer.js**




```
const socket = io('https://localhost:3000', { rejectUnauthorized: false });
let localStream;
const peerConnections = {};
const configuration = {
  iceServers: [{ urls: 'stun:stun.l.google.com:19302' }]
};

async function checkDevices() {
  try {
    const devices = await navigator.mediaDevices.enumerateDevices();
    const hasVideo = devices.some(device => device.kind === 'videoinput');
    const hasAudio = devices.some(device => device.kind === 'audioinput');

    if (!hasVideo) {
      alert('Aucune caméra détectée. Vérifiez votre matériel ou
permissions.');
      return false;
    }
    if (!hasAudio) {
      alert('Aucun microphone détecté. Vérifiez votre matériel ou
permissions.');
      return false;
    }
```

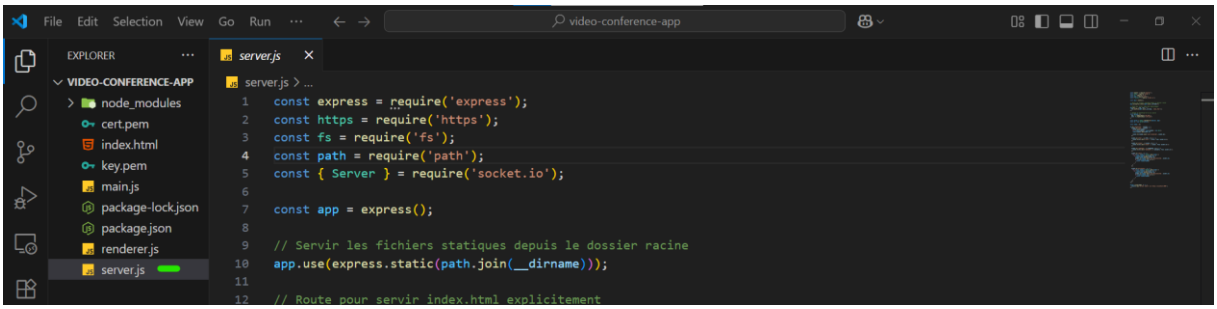
```
    return true;
  } catch (err) {
    console.error('Erreur lors de la vérification des périphériques :', err);
    alert('Erreur lors de la vérification des périphériques : ' +
err.message);
    return false;
  }
}

async function joinRoom() {
  const roomId = document.getElementById('roomId').value;
  if (!roomId) return alert('Veuillez entrer un ID de room');

  // Vérifier les appareils avant de demander l'accès
  const devicesAvailable = await checkDevices();
  if (!devicesAvailable) return;

  try {
    localStream = await navigator.mediaDevices.getUserMedia({
      video: true,
      audio: true
    });
    document.getElementById('localVideo').srcObject = localStream;
    socket.emit('join', roomId);
  } catch (err) {
    console.error('Erreur d’accès à la caméra/micro :', err);
    let errorMessage = 'Erreur d’accès à la caméra/microphone : ';
    if (err.name === 'NotAllowedError') {
      errorMessage += 'Permission refusée. Veuillez autoriser l’accès dans les
paramètres système.';
    } else if (err.name === 'NotFoundError') {
      errorMessage += 'Aucun appareil trouvé. Vérifiez que la
caméra/microphone est connecté.';
    } else if (err.name === 'NotReadableError') {
      errorMessage += 'Appareil déjà utilisé par une autre application.';
    } else {
      errorMessage += err.message;
    }
    alert(errorMessage);
  }
}

function setupPeerConnection(socketId) {
  const pc = new RTCPeerConnection(configuration);
  peerConnections[socketId] = pc;

  localStream.getTracks().forEach(track => {
    pc.addTrack(track, localStream);
  });
```

```
  pc.ontrack = (event) => {
    const video = document.createElement('video');
    video.id = `remoteVideo-${socketId}`;
    video.autoplay = true;
    video.playsinline = true;
    video.srcObject = event.streams[0];
    document.getElementById('videoContainer').appendChild(video);
  };

  pc.onicecandidate = (event) => {
    if (event.candidate) {
      socket.emit('candidate', { candidate: event.candidate, to: socketId });
    }
  };

  return pc;
}

socket.on('user-connected', async (socketId) => {
  const pc = setupPeerConnection(socketId);
  const offer = await pc.createOffer();
  await pc.setLocalDescription(offer);
  socket.emit('offer', { offer, to: socketId });
});

socket.on('offer', async ({ offer, from }) => {
  const pc = setupPeerConnection(from);
  await pc.setRemoteDescription(new RTCSessionDescription(offer));
  const answer = await pc.createAnswer();
  await pc.setLocalDescription(answer);
  socket.emit('answer', { answer, to: from });
});

socket.on('answer', ({ answer, from }) => {
  const pc = peerConnections[from];
  if (pc) {
    pc.setRemoteDescription(new RTCSessionDescription(answer));
  }
});

socket.on('candidate', ({ candidate, from }) => {
  const pc = peerConnections[from];
  if (pc) {
    pc.addIceCandidate(new RTCIceCandidate(candidate));
  }
});

socket.on('user-disconnected', (socketId) => {
```

```
  if (peerConnections[socketId]) {
    peerConnections[socketId].close();
    delete peerConnections[socketId];
    const video = document.getElementById(`remoteVideo-${socketId}`);
    if (video) video.remove();
  }
});

async function shareScreen() {
  try {
    const screenStream = await navigator.mediaDevices.getDisplayMedia({ video:
true });
    const screenTrack = screenStream.getVideoTracks()[0];

    Object.values(peerConnections).forEach(pc => {
      const sender = pc.getSenders().find(s => s.track.kind === 'video');
      if (sender) sender.replaceTrack(screenTrack);
    });

    screenTrack.onended = async () => {
      if (localStream) {
        const videoTrack = localStream.getVideoTracks()[0];
        Object.values(peerConnections).forEach(pc => {
          const sender = pc.getSenders().find(s => s.track.kind === 'video');
          if (sender) sender.replaceTrack(videoTrack);
        });
      }
    };
  } catch (err) {
    console.error('Erreur de partage d’écran :', err);
    alert('Erreur lors du partage d’écran : ' + err.message);
  }
}

function toggleVideo() {
  if (!localStream) return;
  const videoTrack = localStream.getVideoTracks()[0];
  if (videoTrack) {
    videoTrack.enabled = !videoTrack.enabled;
    alert(`Caméra ${videoTrack.enabled ? 'activée' : 'désactivée'}`);
  }
}

function toggleAudio() {
  if (!localStream) return;
  const audioTrack = localStream.getAudioTracks()[0];
  if (audioTrack) {
    audioTrack.enabled = !audioTrack.enabled;
    alert(`Microphone ${audioTrack.enabled ? 'activé' : 'désactivé'}`);
```

```
  }
}

function leaveRoom() {
  Object.values(peerConnections).forEach(pc => pc.close());
  Object.keys(peerConnections).forEach(key => delete peerConnections[key]);

  if (localStream) {
    localStream.getTracks().forEach(track => track.stop());
  }

  const videoContainer = document.getElementById('videoContainer');
  while (videoContainer.children.length > 1) {
    videoContainer.removeChild(videoContainer.lastChild);
  }

  document.getElementById('localVideo').srcObject = null;
  socket.emit('leave');
}
```

Fichier : server.js (serveur de signalisation)




```
const express = require('express');
const https = require('https');
const fs = require('fs');
const path = require('path');
const { Server } = require('socket.io');

const app = express();

// Servir les fichiers statiques depuis le dossier racine
app.use(express.static(path.join(__dirname)));

// Route pour servir index.html explicitement
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'index.html'));
});

// Charger le certificat et la clé privée
const options = {
```

```
  key: fs.readFileSync('key.pem'),
  cert: fs.readFileSync('cert.pem')
};

const server = https.createServer(options, app);
const io = new Server(server);

const rooms = {};

io.on('connection', (socket) => {
  socket.on('join', (roomId) => {
    socket.join(roomId);
    if (!rooms[roomId]) rooms[roomId] = new Set();
    rooms[roomId].add(socket.id);

    socket.to(roomId).emit('user-connected', socket.id);
  });

  socket.on('offer', ({ offer, to }) => {
    socket.to(to).emit('offer', { offer, from: socket.id });
  });

  socket.on('answer', ({ answer, to }) => {
    socket.to(to).emit('answer', { answer, from: socket.id });
  });

  socket.on('candidate', ({ candidate, to }) => {
    socket.to(to).emit('candidate', { candidate, from: socket.id });
  });

  socket.on('leave', () => {
    for (const roomId of Object.keys(rooms)) {
      if (rooms[roomId].has(socket.id)) {
        rooms[roomId].delete(socket.id);
        socket.to(roomId).emit('user-disconnected', socket.id);
        if (rooms[roomId].size === 0) {
          delete rooms[roomId];
        }
      }
    }
  });

  socket.on('disconnect', () => {
    for (const roomId of Object.keys(rooms)) {
      if (rooms[roomId].has(socket.id)) {
        rooms[roomId].delete(socket.id);
        socket.to(roomId).emit('user-disconnected', socket.id);
        if (rooms[roomId].size === 0) {
          delete rooms[roomId];
```

```
        }
      }
    }
  });
});

server.listen(3000, () => {
  console.log('Serveur démarré sur https://localhost:3000');
});
```

6.2 Exécution de l'application

Démarrer le serveur de signalisation

> npm install express socket.io

```
PS E:\video-conference-app> npm install express socket.io

added 78 packages, and audited 161 packages in 8s

30 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS E:\video-conference-app>
```

Dans le dossier de votre projet, exécutez la commande suivante pour générer un certificat et une clé privée

> openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes

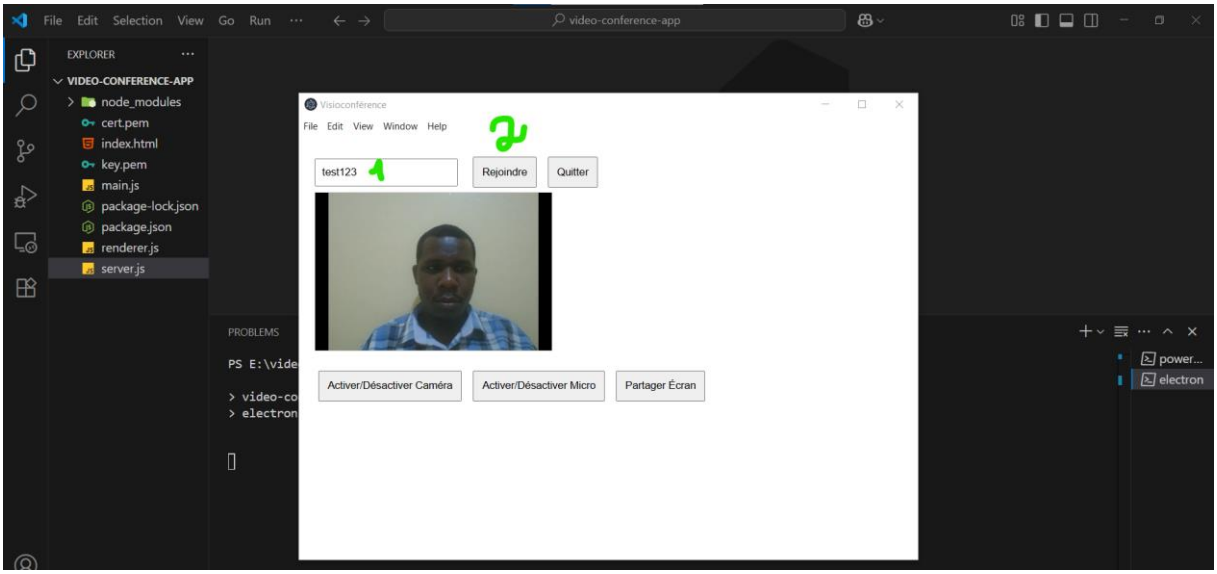
```
Windows PowerShell
PS E:\video-conference-app> openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes
.....+....+.....+.....+.....+...+....+...+.....+.....+..+.+.....+.....+..+.+.....+...+.....+.....++++++++++++++++++++++++++
++++++++++++++++++...+..+.....+...+....+.....+.....+...+.+.....+...+...+.....+.....+++++++++++++++++++++++++++++++++++++++++++++++++...+.....
+....+..+...+.....+.....+....+.....+.....+.....+...+.....+.....+....+++++++++++++++++++++++++++++++++++++++++++++++++.+.+...+...+.....
..+.....+...+.....+.....+.....+.+...+.....+.....+...+....+.....+...+.+.....+.....+.....+.+...+.....+.....+.+.....+...+....
.....+....+..+.+...+.....++++++++++++++++++++++++++++++++++++++++++++++.....+.....+.....+.....+.....+.....+.....+.....+..+.+.....
.....+...+.....+.....+.....+.....+..+.+....+.+.....+.....+.....+.....+.....+.....+.....+.....+...+.+.....
....+..+.....+..+.+.....+.....+.....+.....+...+.....+.....+.....+.....+.....+.....+.....+.....+.....
......+.....+.+.....+.....+.....+.+...+.....+.....+...+.+...+...+.....+...+.....+.....+.....+.....+...+.+...
.....+.....+.+.....+.+.....+.....+....+..+.....+..+.....+.+.....+.....+.....+.....+.....+.....+.....
.+...+.+...+.....+.....+.....+.....+.....+.....+.....+...+.....+.....+.....+.....+.....+.....+.....
.+.....+.....+.....+...+.+.....+.....+.....+.....+.....+.....+.+.+.+.....+...+.+.....+.....+.....
.+..+.....+..+.+.....+.....+....+.....+.....+.+.....+.....+.....+..+.+....+.....+.....+...+.....
+.....+.....+++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:SN
State or Province Name (full name) [Some-State]:Senegal
Locality Name (eg, city) []:Dakar
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ec2lt
Organizational Unit Name (eg, section) []:ec2lt/rtn
Common Name (e.g. server FQDN or YOUR name) []:ec2lt.sn
Email Address []:sergio@gmail.sn
PS E:\video-conference-app>
```

> node server.js

```
PS E:\video-conference-app> node server.js
Serveur démarré sur https://localhost:3000
```

> npm start

```
PS E:\video-conference-app> npm start

> video-conference-app@1.0.0 start
> electron .
```



Rendre l'application accessible sur le réseau local

Modifier l'adresse du serveur dans server.js :

- Remplacez localhost par 0.0.0.0 pour que le serveur écoute sur toutes les interfaces réseau, y compris l'adresse IP locale (192.168.1.3).

- **Mettre à jour l'URL dans renderer.js** :

  - Remplacez https://localhost:3000 par https://192.168.1.3:3000 pour que les clients se connectent à l'adresse IP de la machine hôte.

Mettre à jour main.js :

- Assurez-vous qu'Electron charge l'application depuis https://192.168.1.3:3000 au lieu de localhost.

Gérer les certificats auto-signés :

- Les certificats auto-signés doivent inclure l'adresse IP (192.168.1.3) dans le champ SAN (Subject Alternative Name) pour éviter les erreurs de certificat sur les autres machines.

Étape 1 : Générer un certificat auto-signé avec l'adresse IP

Le certificat actuel (key.pem et cert.pem) doit inclure l'adresse IP 192.168.1.3 dans le champ SAN pour être valide sur le réseau local. Suivez ces étapes pour régénérer le certificat :

Créez un fichier de configuration **openssl.cnf** dans le dossier racine de votre projet avec le contenu suivant :

openssl.cnf

[req]

default_bits = 4096

prompt = no

default_md = sha256

distinguished_name = dn

x509_extensions = v3_req

[dn]

C = SN

ST = Senegal

L = Dakar

O = ec2lt

OU = ec2lt/rtn

CN = localhost

[v3_req]

subjectAltName = @alt_names

[alt_names]

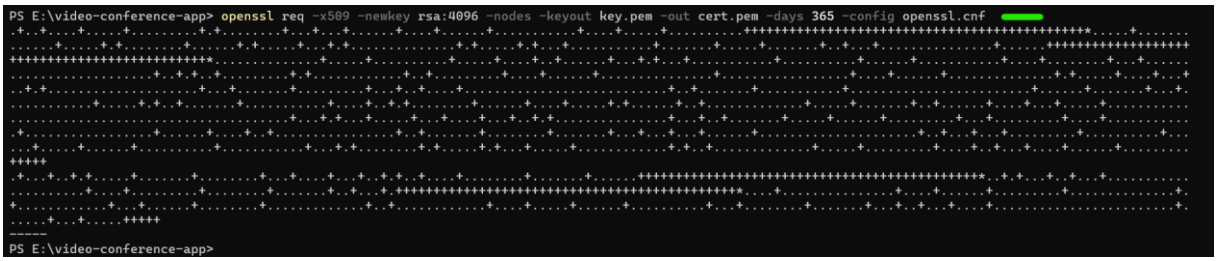
DNS.1 = localhost

IP.1 = 192.168.1.3

email.1 = sergio@gmail.sn

Exécutez la commande suivante pour générer un nouveau certificat et une nouvelle clé privée

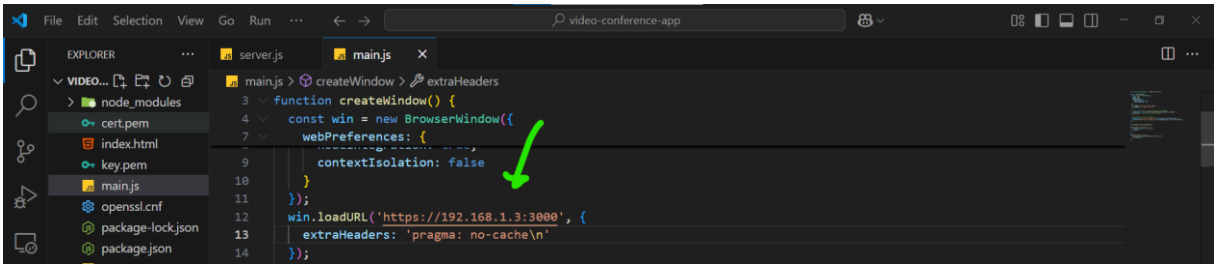
> openssl req -x509 -newkey rsa:4096 -nodes -keyout key.pem -out cert.pem -days 365 -config openssl.cnf



- Modifié **server.listen(3000, ...)** pour inclure **'0.0.0.0'** afin que le serveur écoute sur toutes les interfaces réseau.
- Mis à jour le message de la console pour refléter l'adresse https://192.168.1.3:3000.

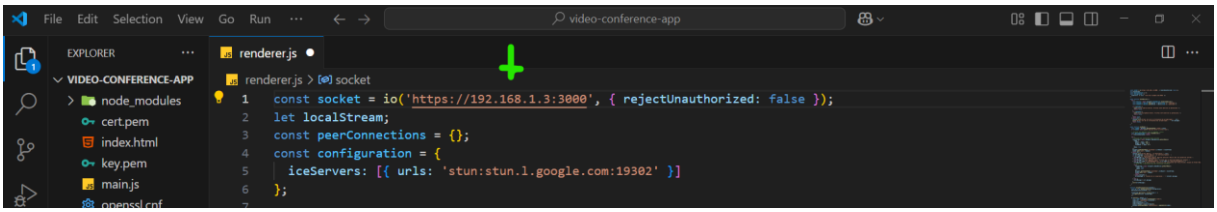
Changements dans main.js :

- Changé l'URL de win.loadURL de https://localhost:3000 à https://192.168.1.3:3000 pour correspondre à l'adresse IP de la machine hôte.



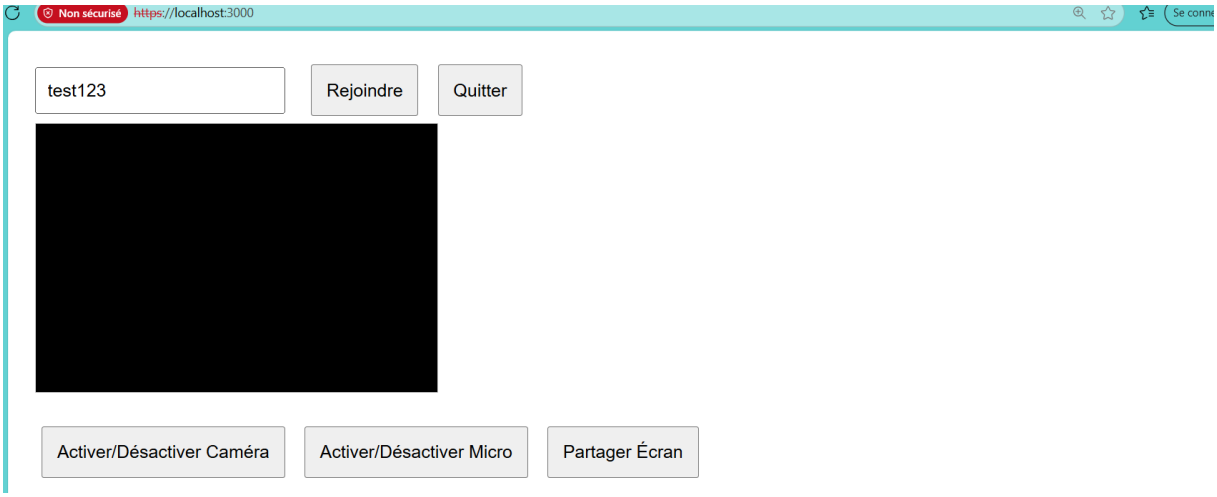
Changements dans renderer.js :

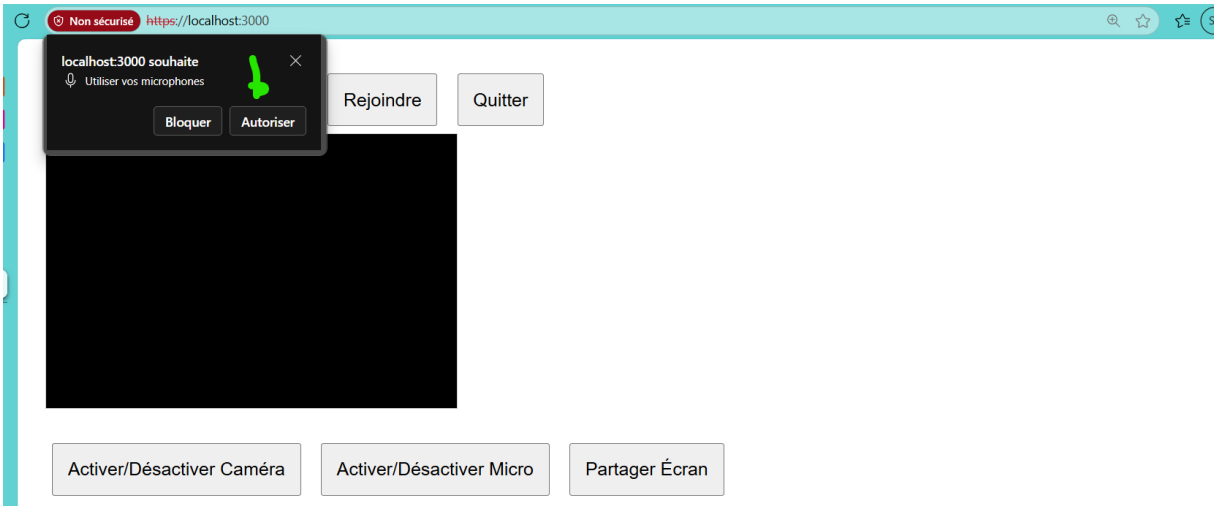
- Changé l'URL de connexion **Socket.IO** de **https://localhost:3000** à **https://192.168.1.3:3000**.
- Conservé l'option **{ rejectUnauthorized: false }** pour accepter le certificat auto-signé.

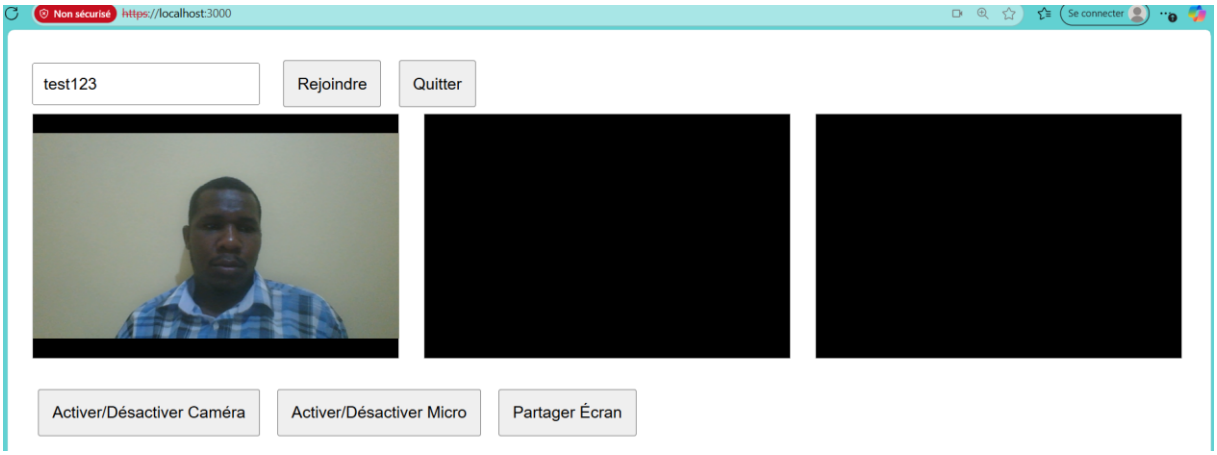


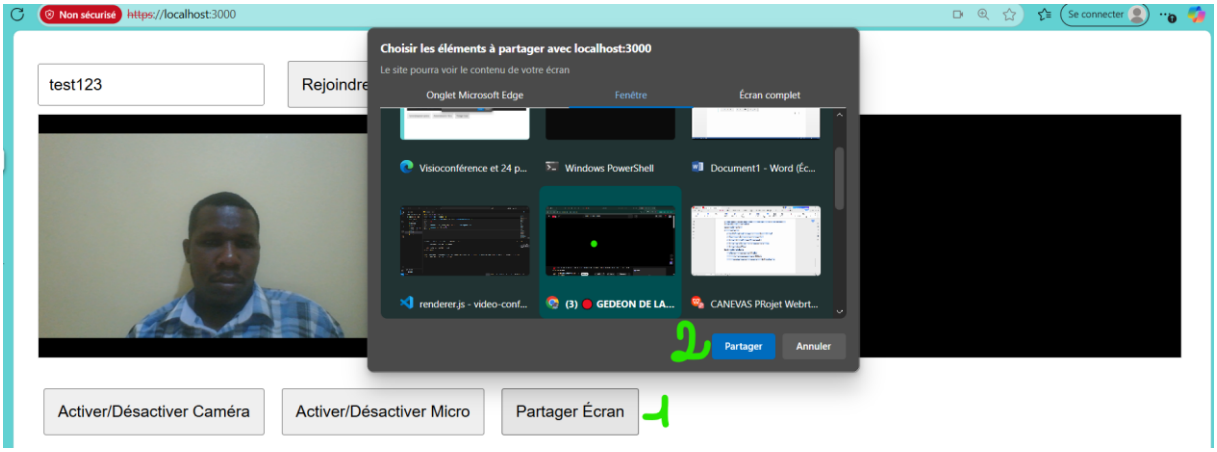
> npm run server

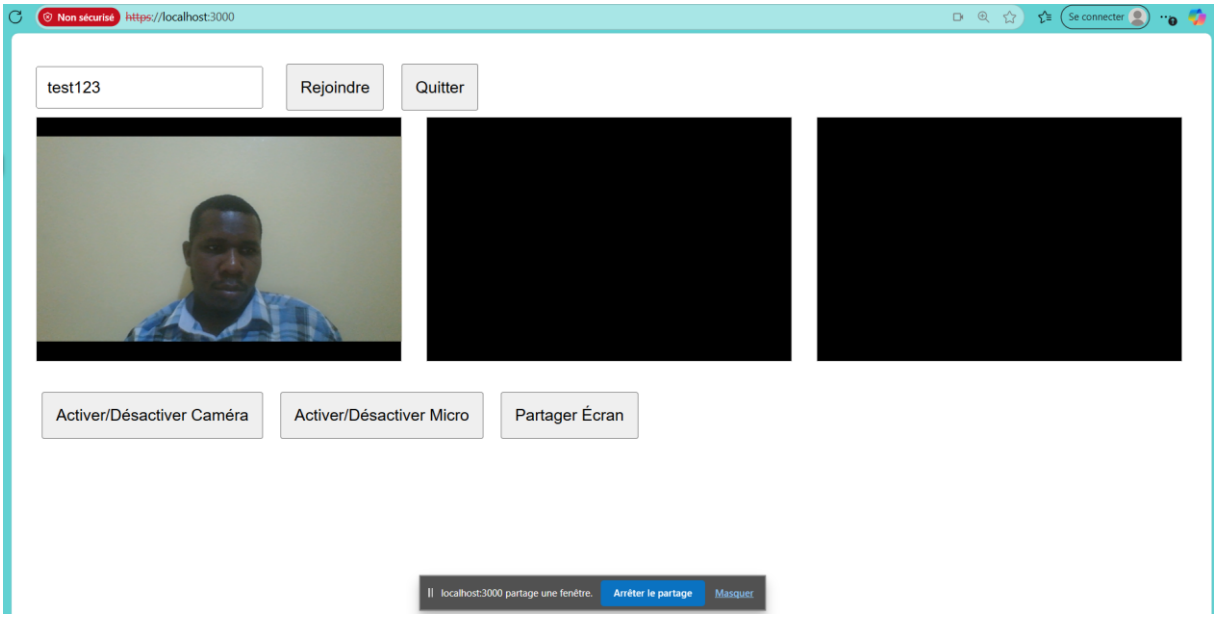
```
PS E:\video-conference-app> npm run server

> video-conference-app@1.0.0 server
> node server.js

Serveur démarré sur https://192.168.1.3:3000
```






Non sécurisé
https://localhost:3000
Se connecter
test123
Rejoindre
Quitter
Activer/Désactiver Caméra
Activer/Désactiver Micro
Partager Écran


Non sécurisé
https://localhost:3000
Se connecter
test123
Choisir les éléments à partager avec localhost:3000
Le site pourra voir le contenu de votre écran
Onglet Microsoft Edge
Fenêtre
Écran complet
Visioconférence et 24 p...
Windows PowerShell
Document1 - Word (Éc...
renderer.js - video-conf...
(3) GEDEON DE LA...
CANEVAS PRojet Webrt...
Partager
Annuler
Activer/Désactiver Caméra
Activer/Désactiver Micro
Partager Écran


Non sécurisé
https://localhost:3000
Se connecter
test123
Rejoindre
Quitter
Activer/Désactiver Caméra
Activer/Désactiver Micro
Partager Écran
localhost:3000 partage une fenêtre.
Arrêter le partage
Masquer

CONCLUSION

Ce projet a permis de concevoir une application de visioconférence fonctionnelle, combinant des technologies de communication en temps réel (WebRTC, PeerJS, Socket.IO) et une interface desktop via Electron. Les fonctionnalités développées, telles que la gestion des connexions P2P, le chat, le partage d'écran et la gestion de comptes avec base de données, répondent aux besoins d'une solution moderne de télécommunication. La génération d'exécutables pour Windows et Linux garantit une accessibilité multiplateforme, validée par des tests rigoureux, offrant ainsi une application fiable et prête à l'emploi.