

# Rapport du projet d'Etude de la technologie WebRTC

Réalisé par :  
Wifkaoui Yasmine  
Taifi Assia  
Al Jadd Mohammed  
El Nabaoui Nouhaila

Encadré par : M. Ibn Elhaj

## **Résumé:**

Les progrès de la technologie ont fourni des potentiels pour de meilleures méthodes de communication. De nouvelles technologies sont apparues pour améliorer les canaux de communication existants. Certaines technologies appliquées à la communication en temps réel présentent plusieurs défis tels que le besoin de plug-ins logiciels supplémentaires et de téléchargements afin d'établir une communication en temps réel, ainsi que des problèmes de sécurité. La communication Web en temps réel (WebRTC) est une technologie qui peut être exploitée pour surmonter ces défis. Le but de cette étude est de comprendre l'architecture de la technologie WebRTC et de développer une application WebRTC pour le chat vidéo.

**Mots-clés :** Technologie, WebRTC, Temps réel, Communication, Sécurité, Application.

## **Abstract :**

Developments in technology have provided potentials for better communication methods. New technologies have emerged to improve existing communication channels. Some technologies applied in Real-time communication come with several challenges such as the need for additional software plugins and downloads in order to establish real-time communication, as well as security issues. Web Real-time Communication (WebRTC) is a technology that can be harnessed to overcome these challenges. The aim of this study is to understand the architecture of WebRTC technology and to develop a WebRTC Application for video chat.

**Key-words:** Technology, WebRTC, Real-time, Communication, Security, Application.

## **Liste des acronymes**

**WebRTC** - Web Real-time Communication.

**SDP** - Session Description Protocol.

**SIP** - Session Initiation Protocol

**NAT** - Network Address Translation.

**STUN** - Session Traversal Utilities for NAT.

**TURN** - Traversal Using Relays around NAT.

**ICE** - Interactive Connectivity Establishment.

**QoE** - Quality of Experience.

**VOIP** - Voice over IP.

**API** - Application programming interface.

**DTLS** - Datagram Transport Layer Security

**RTP** - Real-Time Transport Protocol

**SRTP** - Secure Real-time Transport Protocol

**SCTP** - Stream Control Transmission Protocol

## Sommaire

<b>Résumé:</b> .....	<b>2</b>
<b>Abstract :</b> .....	<b>2</b>
<b>Liste des acronymes</b> .....	<b>3</b>
<b>Introduction</b> .....	<b>5</b>
<b>Problématique</b> .....	<b>6</b>
<b>Objectif</b> .....	<b>6</b>
<b>Chapitre 1 : Etude théorique de la technologie WebRTC</b> .....	<b>7</b>
<b>1  Qu'est-ce que WebRTC :</b> .....	<b>7</b>
<b>2  Description générale de la norme :</b> .....	<b>8</b>
<b>3  Principe de fonctionnement :</b> .....	<b>10</b>
<b>4  Caractéristiques du WebRTC :</b> .....	<b>12</b>
<b>5  Avantages du WebRTC :</b> .....	<b>13</b>
<b>6  Le WebRTC et l'entreprise :</b> .....	<b>14</b>
<b>7  Les applications du web RTC :</b> .....	<b>15</b>
a. WebRTC et l'éducation : .....	15
b. WebRTC et IoT : .....	16
c. WebRTC et santé : .....	17
<b>Chapitre 2 : Etude technique de la technologie WebRTC</b> .....	<b>18</b>
<b>1  L'architecture de la technologie WebRTC :</b> .....	<b>18</b>
<b>2  WebRTC in the Browser :</b> .....	<b>19</b>
<b>3  WebRTC APIs :</b> .....	<b>21</b>
<b>4  Sécurité</b> .....	<b>25</b>
<b>Chapitre 3 : Création d'une Application WebRTC simple</b> .....	<b>26</b>
<b>1  Introduction</b> .....	<b>26</b>
<b>2  Procédure de chat vidéo WebRTC</b> .....	<b>26</b>
<b>3  Préparation de notre environnement</b> .....	<b>27</b>
a. Les technologies utilisées .....	27
b. La configuration de Firebase .....	28
<b>4  Explication du code de notre Application</b> .....	<b>32</b>
a. Partie HTML (index.html) : .....	32
b. Partie CSS (style.css) : .....	32
c. Partie Javascript (script.js) : .....	33
<b>5  Test</b> .....	<b>35</b>
<b>6  Conclusion</b> .....	<b>36</b>
<b>Conclusion générale</b> .....	<b>37</b>
<b>Références</b> .....	<b>38</b>

# **Introduction**

Internet n'est pas étranger à l'audio et à la vidéo. Les applications Web de tous les jours, telles que Netflix et Pandora, diffusent du contenu audio et vidéo à des millions de personnes. En revanche, le Web est étranger à la communication en temps réel. Les sites Web, tels que Facebook, commencent tout juste à activer la communication vidéo dans un navigateur, et ils utilisent généralement un plugin que les utilisateurs doivent installer. C'est là qu'intervient Web Real Time Communication (WebRTC).

La communication vidéo devient également aussi répandue que la communication audio. Avec des technologies telles que FaceTime d'Apple, Google Hangouts et les appels vidéo Skype, parler à quelqu'un via un flux vidéo est une tâche simple pour un utilisateur quotidien.

Un large éventail de techniques a été développé dans ces applications pour garantir que la qualité de la vidéo est une excellente expérience pour l'utilisateur. Il existe des solutions techniques à des problèmes, tels que la perte de paquets de données, la récupération après des déconnexions et la réaction aux changements dans le réseau d'un utilisateur.

WebRTC exploite un ensemble d'API sans plug-in pouvant être utilisées à la fois dans les navigateurs de bureau et mobiles, et est progressivement pris en charge par tous les principaux fournisseurs de navigateurs modernes. Auparavant, des plugins externes étaient nécessaires pour obtenir des fonctionnalités similaires à celles offertes par WebRTC.

WebRTC exploite plusieurs normes et protocoles, dont la plupart seront abordés dans ce rapport. Ceux-ci incluent les flux de données, les serveurs STUN / TURN, la signalisation, JSEP, ICE, SIP, SDP, NAT, UDP / TCP, les sockets réseau, etc.

C'est dans ce contexte que s'inscrit notre travail, qui dévoilera le rôle important du WebRTC pour la communication en temps réel dans le web.

Pour arriver à mieux cerner la problématique, on a subdivisé notre travail en trois grandes parties :

Dans la première partie, on va introduire, avec plus de détails, une description de la problématique traitée ainsi que l'objectif principal de notre projet avec une étude théorique de la technologie WebRTC, puis dans la deuxième partie, on va faire une étude technique, pour passer à la troisième partie, à la réalisation de l'application simple WebRTC de chat Vidéo et enfin une simulation.

## **Problématique**

D'après le site Internet Live Stats, le monde comptait 4,79 milliards d'internautes en 2020. l'usage des smartphones et des tablettes est en croissance considérable.

L'existence d'une panoplie de protocoles de communication ont conduit à la diversité des logiciels de chat.

Cependant, l'absence de la possibilité d'échanger des textes et de faire des appels vidéo entre différentes applications a posé un vrai problème. Puisque, Vous devez être sûr que tous les participants, qu'il s'agisse d'amis, de parents ou de collègues, ont la même application de communication, et vous devez télécharger et installer une nouvelle version de cette application chaque fois qu'un développeur apporte quelques modifications aux protocoles de communication. Eh bien, c'est là que la technologie WebRTC intervient une véritable révolution du web et de la communication en temps réel.

Alors, qu'est-ce que cette nouvelle technologie ? comment fonctionne-t-elle ? qu'elles sont ces bénéfices pour l'entreprise ?... Et plein d'autres questions que nous allons essayer de répondre durant notre présentation.

## **Objectif**

Le but principal de notre travail est d'étudier d'une manière générale la technologie WebRTC tout en abordant sa partie théorique, technique et pratique.

# **Chapitre 1 : Etude théorique de la technologie WebRTC**

## **1 | Qu'est-ce que WebRTC :**



L'histoire du WebRTC a débuté en Mai 2010 avec le rachat de la société Global IP Solutions (GIPS) par Google. L'objectif de ce rachat est plus technologique que business. En effet, un an après le rachat de GIPS, Google publie sous licence open source les technologies issues de ce rachat, servant de fondation au projet WebRTC.

Dès 2011, Google s'associe à différents éditeurs comme Mozilla et Opéra ainsi qu'aux organismes de standardisations de W3C et de l'IETF pour développer le WebRTC et en faire un standard technologique ouvert.

Le WebRTC est ainsi devenu le nouveau standard de communication peer-to-peer entre navigateurs web. Il permet notamment à tout internaute disposant d'une application web VoIP compatible, de passer des appels ou de réaliser des visio-conférences directement au sein de son navigateur. Plus besoin d'extension, de widgets ou de plugins, tout est 100% natif au navigateur.

Alors vous nous direz que la communication en temps réel entre navigateurs web n'est pas nouvelle. Et bien vous auriez raison. La première nouveauté du WebRTC est d'offrir un standard ouvert, à l'instar des solutions et des plugins propriétaires existants. Le WebRTC doit ainsi faciliter l'implémentation et l'adoption de cet usage, tout en garantissant l'interopérabilité potentielle des solutions.

Les principes fondateurs du projet WebRTC sont que ses APIs doivent être open source, gratuites, standardisées, compatibles avec les principaux navigateurs, tout en offrant une couverture fonctionnelle supérieure aux solutions existantes.

Le WebRTC est aujourd'hui pleinement intégré aux navigateurs Google Chrome, Mozilla Firefox et Opera. Il est aujourd'hui utilisé dans des usages aussi variés que les jeux vidéo, le streaming, la messagerie instantanée ou toute autre application nécessitant de la communication en temps réel.

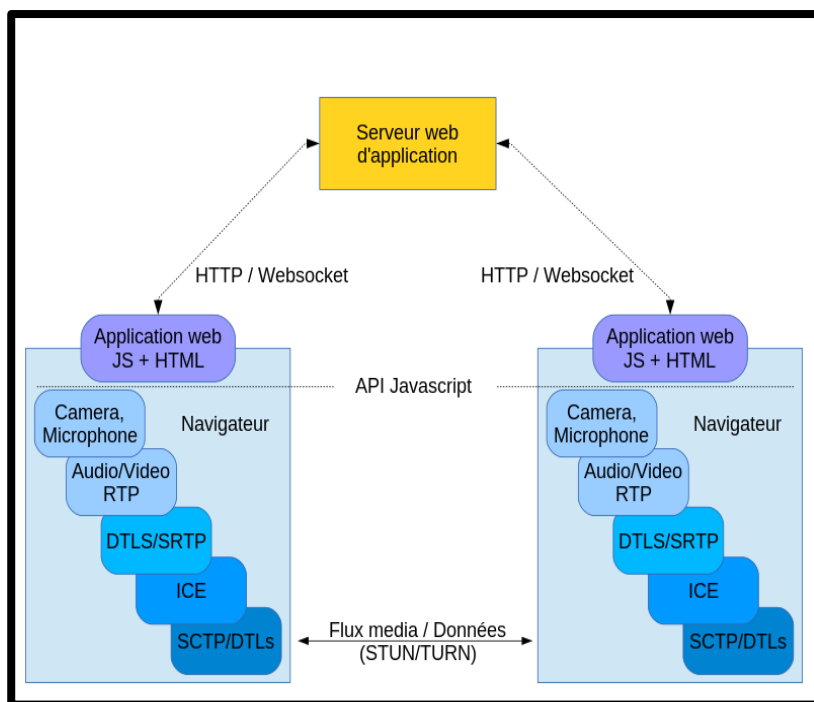
## 2| Description générale de la norme :

L'architecture de l'API WebRTC repose sur une construction triangulaire impliquant un serveur et deux pairs.

Les deux navigateurs téléchargent depuis un serveur une application JavaScript vers leur contexte local. Le serveur est utilisé comme point de rendez-vous afin de coordonner les échanges entre navigateurs jusqu'à ce que la connexion directe entre navigateurs soit établie.

L'application téléchargée utilise l'API WebRTC pour communiquer avec le contexte local.

Comment nous voyons dans l'image en dessus, le but est d'avoir une application cliente en JavaScript et HTML5 interagissant avec le navigateur au travers de l'API WebRTC.



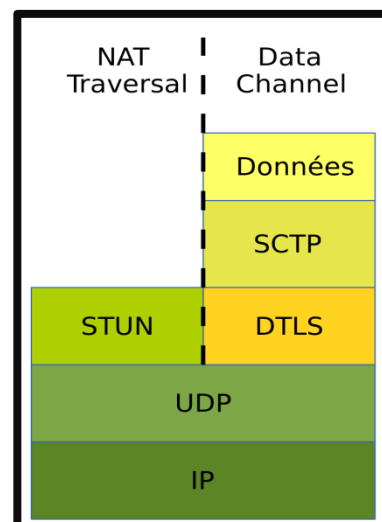
**WebRTC** permet aux applications et sites Web de capturer et éventuellement de diffuser des médias audios et / ou vidéo, ainsi que d'échanger des données arbitraires entre les navigateurs sans passer par un intermédiaire. L'ensemble de normes qui composent WebRTC permet de partager des données et d'effectuer des téléconférences peer-to-peer, sans exiger que l'utilisateur installe des plug-ins ou tout autre logiciel tiers.

**WebRTC** permet l'échange de données telles que des images ou du texte en utilisant la composante Data Channel. Dans ce qui suit, nous décrirons ce qu'est le canal de données.

- **Data channels :**

L'API DATA channels offre un moyen d'échange de données génériques bidirectionnel et pair à pair. Cette composante de WebRTC permet l'échange de données telles que des images ou du texte. Une première démonstration par Mozilla a eu lieu en novembre 2012.

Ces canaux de données sont créés entre pairs en utilisant l'objet PeerConnection. Ces données autres que les flux médias sont échangées via le protocole SCTP, lui-même encapsulé dans DTLS. Cette solution permet au flux de données d'être intégré dans le même paquet que les flux de médias et donc de partager le même numéro de port pour les échanges.





**SCTP** supporte nativement plusieurs flux de données de façon bidirectionnelle (jusqu'à 65536 dans chaque direction) au sein d'une association SCTP et gère les priorités. De cette façon, il est possible de favoriser les messages de haute priorité face aux gros objets à priorité basse. Chaque flux représente une connexion logique unidirectionnelle.

Afin d'assurer la confidentialité et l'authenticité des paquets SCTP échangés, chaque flux repose sur le protocole **DTLS**.

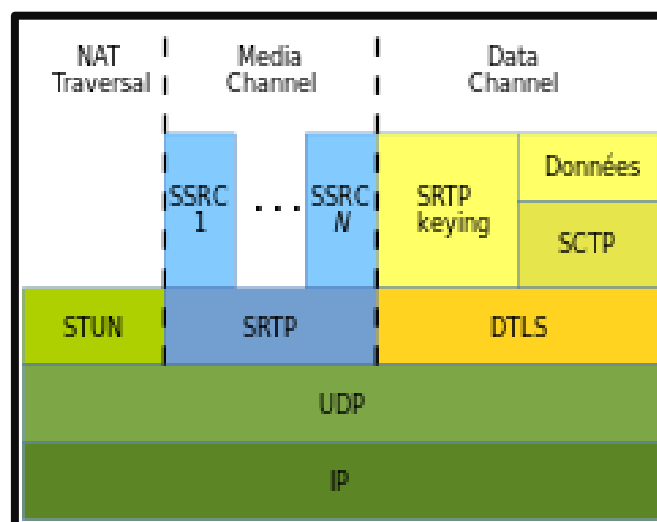
Au sein d'un canal de données, les applications peuvent transmettre des messages de façon ordonnée ou désordonnée. L'ordre de remise est préservé uniquement dans le cas d'une transmission de paquets ordonnés envoyés sur le même lien de données.

Un flux de données est créé lorsque l'un des pairs appelle une méthode **CreateDataChannel()** pour la première fois après avoir créé un objet PeerConnection. Chaque appel suivant à **CreateDataChannel()** créera un nouveau flux de données au sein de la connexion SCTP existante.

Le protocole DTLS n'a pas pour seul rôle d'encapsuler les paquets SCTP. Dans le cadre d'un multiplexage avec des flux médias, le protocole DTLS encapsule la gestion des clés et la négociation des paramètres pour le protocole **SRTP**, utilisé pour la gestion des flux médias. Il y a donc dépendance du flux média vis-à-vis du flux de données

- **Flux Media :**

Un MediaStream est une représentation d'un flux de données spécifique audio ou vidéo. Il permet la prise en charge des actions sur le flux média telles que l'affichage, l'enregistrement et l'envoi à un pair distant. Un **MediaStream** peut être local ou distant. L'API MediaStream gère les flux audio et vidéo et indique à l'application qu'elle doit donner accès à la caméra, aux haut-parleurs et au microphone ; Afin d'être utilisé, un MediaStream local doit demander l'accès aux ressources multimédia de l'utilisateur via la fonction *getUserMedia()*.



L'application spécifie le type de média (audio ou vidéo) auquel elle souhaite accéder et le navigateur autorise ou refuse l'accès à la ressource demandée. Une fois que le média n'est plus utilisé, l'application peut révoquer son propre accès avec la méthode stop() sur le flux média local.

Les flux médias sont transportés par le biais du protocole RTP, utilisable sur tout protocole de transport implémentant une abstraction de datagram (UDP par exemple). La confidentialité, l'authentification des messages et la protection contre les répétitions sont apportées par l'utilisation sécurisée de RTP, SRTP.

La gestion des clés pour SRTP est assurée par DTLS et donc le flux de données. Il est donc impossible d'avoir un flux média indépendant d'un flux de données là où l'inverse est envisageable.

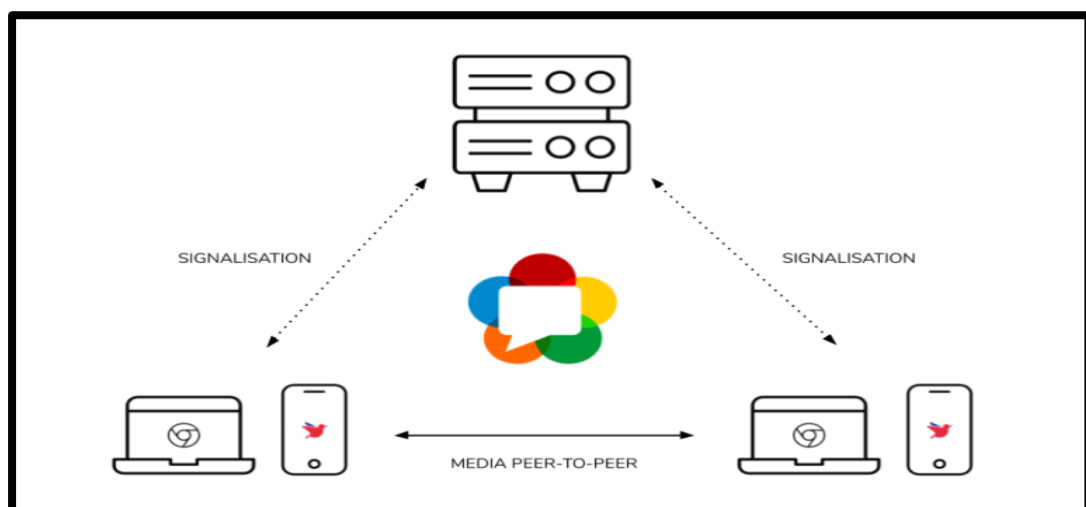
Il est possible d'associer plusieurs flux médias sur une même connexion SRTP qui utiliseront des ressources médias différentes ou non. Dans ce cas, les sources de chaque flux sont clairement identifiées comme des SSRC.

### **3| Principe de fonctionnement :**

Le WebRTC fonctionne en peer-to-peer. Le flux média ne transite (en théorie) donc pas par un serveur centralisé mais directement entre les navigateurs des utilisateurs. Le serveur centralisé se chargeant simplement de la signalisation et de la gestion de la mise en relation entre les utilisateurs.

La signalisation et le média sont donc gérés différemment dans le WebRTC. La signalisation se fait généralement au sein d'une WebSocket mais cette dernière n'est pas définie par le WebRTC. Vous êtes donc libre de choisir le protocole de signalisation le plus adapté à votre média, que ce soit le protocole SIP ou XMPP par exemple.

Le média quant à lui sera géré dans un chemin différent utilisant le média channel et le protocole SRTP pour la voix et la vidéo ou le data channel et le protocole SCTP pour les données.

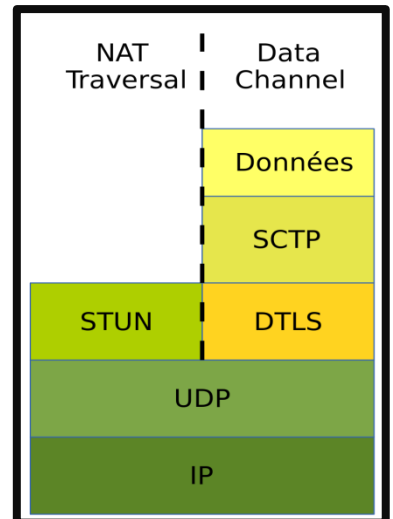


Lorsque la demande de connexion est initiée par un navigateur, une négociation ICE, pour Interactive Connectivity Establishment, est initiée. Elle est en charge d'identifier si la connexion directe est possible entre les navigateurs ou si il faut avoir recours à un relai. Cette procédure a été récemment améliorée par Trickle ICE, afin d'en améliorer l'efficacité.

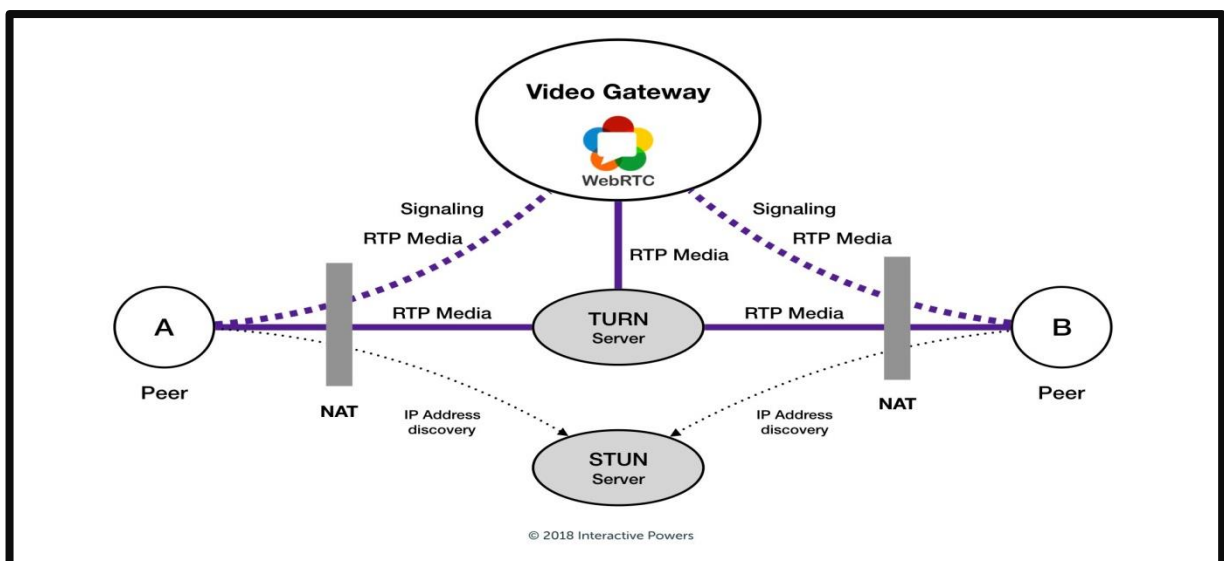
Dans le cadre d'un réseau utilisant de la translation d'adresses (Network Address Translation ou NAT), il convient également d'utiliser le protocole STUN, pour Session Traversal Utilities for NAT.

Il aura la charge d'identifier l'adresse IP et le port de connexion de la personne à connecter se trouvant au sein d'un réseau privé.

Au sein de certains réseaux très sécurisés et restrictifs, il se peut que la connexion ne puisse se faire directement entre navigateurs. L'utilisation d'un serveur relai est alors nécessaire. Ce composant de l'architecture d'une installation de WebRTC est appelé TURN, pour Traversal Using Relays around NAT. Il est estimé à près de 20% le nombre de sessions nécessitant l'utilisation d'un serveur TURN.



### • TURN vs STUN :



L'utilisation du protocole appelé STUN (Session Traversal Utilities for NAT) permet aux clients de découvrir leur adresse IP publique et le type de NAT derrière eux. Ces informations sont utilisées pour établir la connexion multimédia. Dans la plupart des cas, un serveur STUN n'est utilisé que pendant la configuration de la connexion et une fois que cette session a été établie, les médias circuleront directement entre l'homologue et la passerelle vidéo (WebRTC).

Cependant, même si nous configurons correctement un serveur STUN, il existe des réseaux d'entreprise très restrictifs (ex: trafic UDP interdit, seulement 443 TCP autorisés...), ce qui obligera les clients à utiliser un serveur TURN (Traversal Using Relays around NAT) pour relayer le trafic si la connexion directe (pair à la passerelle vidéo) échoue. Dans ces cas, vous pouvez installer notre serveur TURN (dans un autre cas) pour résoudre ces problèmes.

#### **4| Caractéristiques du WebRTC :**



Le développement rapide de la technologie dans le domaine des télécommunications et la forte compétitivité sur le marché due aux exigences des consommateurs incite les entreprises à acquérir les meilleurs outils technologiques pour fidéliser leur clientèle. Une entreprise doit assurer un service rapide, efficace et simple sans augmenter ses coûts. Dans l'évolution de ces outils technologiques appliqués aux télécommunications, le WebRTC se présente comme un nouveau standard utilisé pour répondre aux exigences des utilisateurs en effet le web RTC propose diverses fonctionnalités :

- Messages d'accueil
- Menu interactif vocal
- Enregistrement des appels
- Horaires
- Transferts d'appels
- Files d'attente
- Boîte vocale
- Identification des appels
- Statistiques
- Renvoi sans limite de minutes
- Filtration par origine d'appel
- Listes blanches ou noires

## 5| Avantages du WebRTC :

Le WebRTC est une solution très intéressante pour la télécommunication elle présente divers avantages :

- Le WebRTC ne nécessite pas des équipements ou téléphone IP ainsi aucun frais d'installation et d'entretien n'est nécessaire.
- C'est une technologie sans investissement vous n'avez pas besoin d'investissement dans du matériel informatique spécial ou des équipements divers.
- Pas d'achat de licence (Opus et VP8).
- De plus Le WebRTC, pour son installation exige uniquement un navigateur Web qui soutienne la technologie WebRTC et l'autorisation d'accéder à la webcam, au microphone et au haut-parleur de l'ordinateur.



- C'est une technologie qui fonctionne avec n'importe quel navigateur disposant d'une connexion à Internet via data ou wifi.
- Une grande **amélioration dans la qualité de la voix** et de la vidéo : Google affirme qu'il a une grande **amélioration dans la qualité de la voix** et de la vidéo en parle de voix HD.
- De plus, le WebRTC s'adapte aux variations du réseau, en offrant la meilleure qualité possible à chaque instant, même lorsque la connexion est faible, contrairement aux services précédents de téléphonie IP.
- Une grande flexibilité et mobilité : Vous pouvez l'utiliser sur votre téléphone portable, votre ordinateur ou votre tablette en fonction de votre disponibilité et pour votre commodité.

## **6| Le WebRTC et l'entreprise :**

- Le développement de la technologie WebRTC a représenté une grande avancée dans le secteur des télécommunications, et présente de nombreux avantages pour tous types d'entreprises, qu'elles soient petites ou grandes.
- **Accessible à tous et partout :**  
La gestion de la communication d'entreprise était auparavant un travail compliqué et coûteux. Aujourd'hui, avec les nouvelles technologies, il est devenu de plus en plus facile et accessible à tous, sans besoin d'effectuer de grosses dépenses. Le WebRTC fait partie de ces nouvelles technologies et a donc révolutionné les communications internationales. Il est maintenant possible de mettre en communication deux personnes en temps réel, via un simple navigateur connecté à internet.
- **Une flexibilité d'utilisation :**  
Les appels peuvent être passés depuis un ordinateur, depuis un téléphone portable ou une tablette. Peu importe, la qualité reste la même et l'interlocuteur ne verra jamais la différence.
- **Une adaptabilité continue au réseau :**  
Le WebRTC s'adapte aux variations du réseau, ce qui lui permet d'offrir la meilleure qualité de communication possible à chaque instant, même lorsque la connexion est faible. Le WebRTC se caractérise donc par une grande adaptabilité aux variations de qualité du réseau, ce qui élimine les coupures et les échos habituels. Le logiciel est parfait pour appeler de n'importe quel pays et vers n'importe quelle destination : la voix est transmise via Internet, en qualité HD, sans aucun coût supplémentaire.
- **Une image professionnelle :**  
Un standard téléphonique virtuel doté de la technologie WebRTC permet d'améliorer la performance des communications et d'offrir un service de qualité aux clients grâce à une meilleure gestion des appels.
- **Mobilité totale :**  
Lorsqu'il s'agit d'appeler via Internet, les barrières géographiques disparaissent. Travailler avec un standard téléphonique WebRTC signifie pouvoir répondre aux appels de l'entreprise à tout moment, où que l'on soit sans avoir à donner de numéro de téléphone différent à ses clients. Pour cette raison, le WebRTC permet une mobilité totale dans le monde entier.
- **Accès à des statistiques :**  
La téléphonie virtuelle offre, parmi ses multiples fonctionnalités, le service de statistiques avancées, qui collecte différents types de données sur les appels qui ont eu lieu, créant ainsi une base de données qui fournit des informations très précieuses pour l'entreprise et son activité.
- **Fonctions avancées :**  
Contrairement à la téléphonie traditionnelle, qui entraîne généralement des frais ou des charges supplémentaires, la téléphonie virtuelle inclut dans ses tarifs une série de fonctionnalités avancées utiles pour tout type d'entreprise, petite ou grande. Les

appels sont passés et reçus comme d'habitude mais le standard virtuel offre un plus grand nombre de possibilité

- **Cout diminué :**

La plate-forme de communication intégrée de WebRTC permet aux entreprises avec de petits budgets d'améliorer leur connectivité. En raison de sa nature open source et de l'API standardisée, le WebRTC favorise le développement rapide d'applications, ainsi que la réduction du temps nécessaire pour apprendre et mettre en œuvre des solutions basées sur le Web. Ainsi, sur le long terme, les entreprises dépensent moins d'argent sur les infrastructures et consacrent plus d'argent à ce qui est essentiel à leur croissance. Des lignes fixes peuvent être coûteuses et peu pratiques en raison de l'installation de l'équipement et du coût par minute. Tandis qu'avec un système de VoIP et de WebRTC, le seul matériel nécessaire est celui que vous avez déjà.

## 7| **Les applications du web RTC :**

### a. **WebRTC et l'éducation :**



- On peut tirer de la valeur ajoutée de cette des technologies afin de pouvoir améliorer l'éducationnel effet WebRTC permet de concevoir et de créer des outils d'apprentissage en ligne personnalisés, interactifs et intuitifs pour tous les groupes d'âge. Par exemple discours et conférences pour l'école élémentaire, collège notamment les universités.
- La mise en place d'un service de Conseils aux étudiants ainsi d'Autres Services aux étudiants, comme l'aide financière et la planification de carrière, Inscription et orientation, événements à distance, rassemblements et réunions et activités de club parascolaires.



### Pourquoi ce choix :

- Ses mesures de cryptage et de sécurité en font la technologie vidéo et audio la plus sécurisée du marché. Contrairement aux autres outils d'apprentissage à distance qui n'utilisent pas WebRTC, vous pouvez être assuré que vos élèves et vos salles de classe restent en sécurité et protégés.

Vous pouvez utiliser n'importe quel appareil. Les étudiants et le personnel peuvent utiliser un ordinateur de bureau, un ordinateur portable, une tablette ou un Smartphone. Avec un navigateur Web et une connexion Internet ou cellulaire, ils sont prêts à fonctionner.

### **b. WebRTC et IoT :**



- Il existe un certain nombre de domaines où l'intersection entre le périphérique IoT et la communication WebRTC crée des opportunités pour une intégration améliorée. Par exemple, une gamme de produits de sécurité domestique qui, utilisent des capteurs pour détecter quand quelqu'un est à la porte. Plutôt que de simplement prendre une photo ou d'alerter le propriétaire lorsque le capteur est déclenché, le capteur HD intègre WebRTC pour offrir un streaming vidéo, permettant au propriétaire de voir exactement ce qui se passe.



### c. WebRTC et santé :



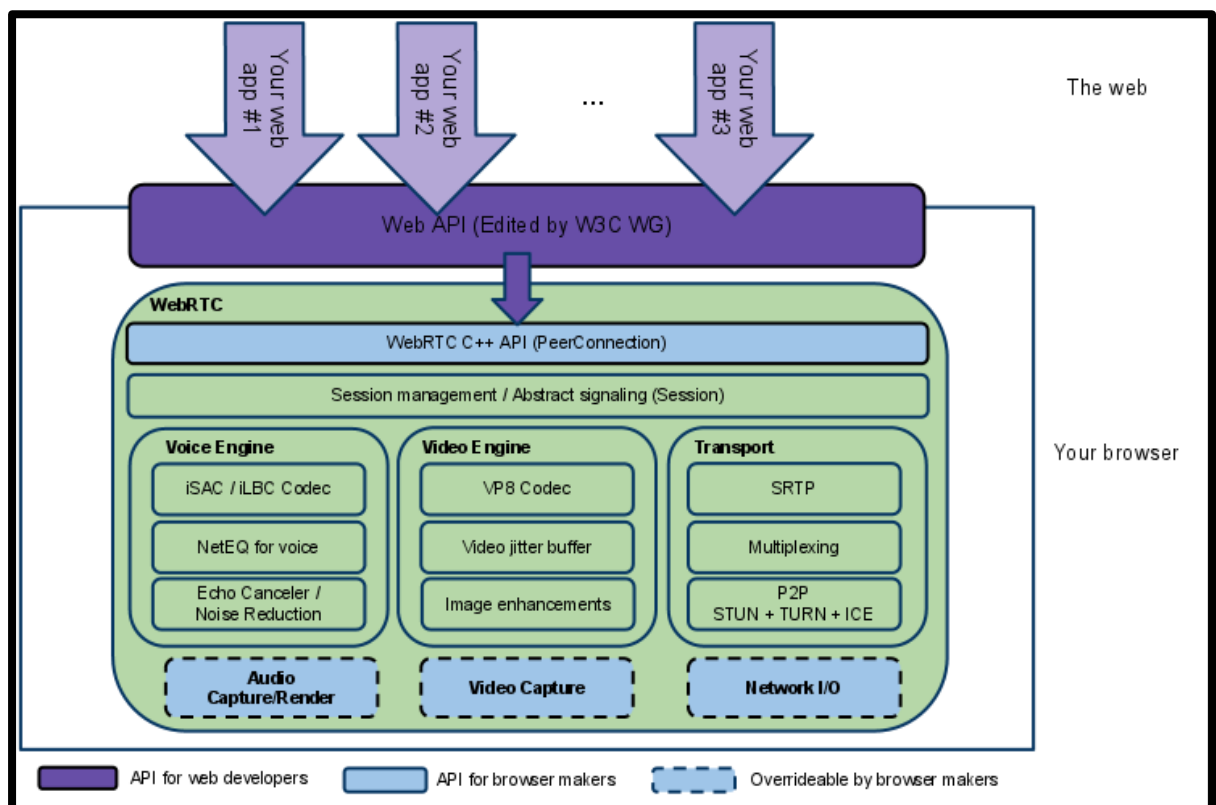
- WebRTC permet une nouvelle génération d'applications de télésanté. La télésanté est une partie en pleine expansion des soins de santé à l'échelle mondiale, avec le potentiel de réduire les coûts et de mieux servir les patients avec des soins spécialisés, quel que soit leur lieu de résidence.

## Chapitre 2 : Etude technique de la technologie WebRTC

### 1| L'architecture de la technologie WebRTC :

On commence par l'Explication de l'identification des couleurs du diagramme d'architecture :

- **La partie Violet** est la couche API développeur web.
- **La ligne bleue solide** est la couche API pour les fabricants de navigateurs (c'est-à-dire le module à l'intérieur de la boîte rouge).
- **Une partie de la ligne pointillée bleue** peut être personnalisée par les fabricants de navigateurs.



On passe aux composants d'architecture WebRTC :

- **Votre application Web** : Un programme développé par un développeur web.
- **Web API** : L'API standard WebRTC (Javascript) pour les développeurs tiers permet aux développeurs de développer facilement des applications Web similaires au chat vidéo en ligne.
- **WebRTC Native C++ API** : La couche locale d'API C++ permet aux fabricants de navigateurs de mettre en œuvre l'API Web standard WebRTC et de traiter les processus de signaux numériques de manière abstraite.
- **Transport / Session** : Les composants de session sont construits en réutilisant les composants de libjingle, sans utiliser ou exiger le protocole xmpp/jingle.
- **Pile RTP** : Une pile réseau pour RTP, le protocole en temps réel.
- **Etourdissement/Glace** : Un composant permettant aux appels d'utiliser les

mécanismes STUN et ICE pour établir des connexions entre différents types de réseaux.

- **Gestion des séances** : Une couche de session abstraite, permettant la configuration des appels et la couche de gestion. Cela laisse la décision de mise en œuvre du protocole au développeur de l'application.
- **Voice Engine** : VoiceEngine est un cadre pour la chaîne de médias audio, de la carte son au réseau.
- **iSAC**: Un codec audio large bande et super large bande pour VoIP et audio en streaming. iSAC utilise une fréquence d'échantillonnage de 16 kHz ou 32 kHz avec un taux de bits adaptatif et variable de 12 à 52 kbps.
- **iLBC**: Un codec de la parole à bande étroite pour la VoIP et le streaming audio. Utilise une fréquence d'échantillonnage de 8 kHz avec un bitrate de 15,2 kbps pour les cadres de 20 ms et de 13,33 kbps pour les cadres de 30 ms. Défini par les CRF IETF 3951 et 3952.
- **Opus** : Prend en charge l'encodage constant et variable des bitrates de 6 kbit/s à 510 kbit/s, la taille des images de 2,5 ms à 60 ms, et divers taux d'échantillonnage de 8 kHz (avec 4 kHz de bande passante) à 48 kHz (avec bande passante de 20 kHz, où toute la portée auditive du système auditif humain peut être reproduite).
- **Acoustic Echo Canceled (AEC)** : L'Acoustic Echo Canceled est un composant de traitement de signal basé sur un logiciel qui supprime, en temps réel, l'écho acoustique résultant de la voix jouée à venir dans le microphone actif.
- **Réduction du bruit (NR)** : Le composant réduction du bruit est un composant de traitement de signal basé sur un logiciel qui supprime certains types de bruit de fond habituellement associés à la VoIP. (Sœurs, bruit de ventilateur, etc....).
- **Vidéo Engine** : VideoEngine est une chaîne de médias vidéo cadre pour la vidéo, de la caméra au réseau, et du réseau à l'écran.
- **VP8 (VP8)** : Codec vidéo du projet WebM. Bien adapté pour le RTC car il est conçu pour une faible latence.
- **Vidéo Jitter Buffer** : Dynamic Jitter Buffer pour la vidéo. Aide à dissimuler les effets de la nervosité et de la perte de paquets sur la qualité globale de la vidéo.
- **Améliorations d'image** : Par exemple, supprime le bruit vidéo de la capture d'image par la webcam.

## 2| WebRTC in the Browser :

Le Web va si vite et il s'améliore constamment. De nouvelles normes sont créées chaque jour. Les navigateurs permettent d'installer des mises à jour sans que l'utilisateur ne le sache, on doit donc suivre ce qui se passe dans le monde du Web et du WebRTC.

Chaque navigateur n'a pas toutes les mêmes fonctionnalités WebRTC en même temps. Différents navigateurs peuvent être en avance sur la courbe, ce qui rend certaines fonctionnalités WebRTC travailler dans un navigateur et non pas un autre.

- **Chrome, Firefox et Opera**

Les dernières versions de Chrome, Firefox et Opera sur les systèmes d'exploitation PC grand public tels que Mac OS X, Windows et Linux, tous soutiennent WebRTC out-of-the-box. Et surtout, les ingénieurs des équipes de développeurs Chrome et Firefox ont travaillé ensemble pour résoudre les problèmes afin que ces deux navigateurs puissent communiquer facilement les uns avec les autres.

- **Android OS**

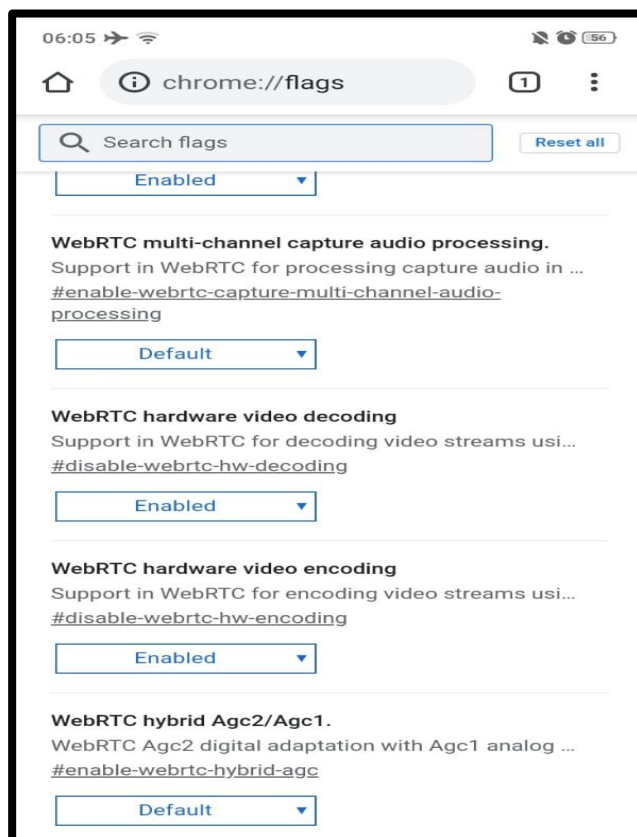
Sur les systèmes d'exploitation Android, les applications WebRTC pour Chrome et Firefox devraient fonctionner en dehors de la boîte. Ils sont en mesure de travailler avec d'autres navigateurs après Android Ice Cream Sandwich version (4.0). Cela est dû au partage de code entre les versions de bureau et mobiles.

- **Apple**

Apple n'a pas encore fait d'annonce sur leurs plans pour soutenir WebRTC dans Safari sur OS X. L'une des solutions de contournement possibles pour les applications iOS natives hybrides est de pour intégrer le code WebRTC directement dans l'application et charger cette application dans un WebView.

Voici un exemple Pour activer WebRTC sur Chrome pour Android :

1. Tapez chrome : // flags / dans l'omnibox pour accéder aux indicateurs.
2. Faites défiler d'environ un tiers vers le bas et activez l'indicateur Activer WebRTC.
3. Il vous sera demandé de relancer le navigateur en bas de la page pour que le drapeau prenne effet.



### 3| WebRTC APIs :

Il existe trois API principales qui gèrent la capture audio, la visioconférence et la transmission de données :

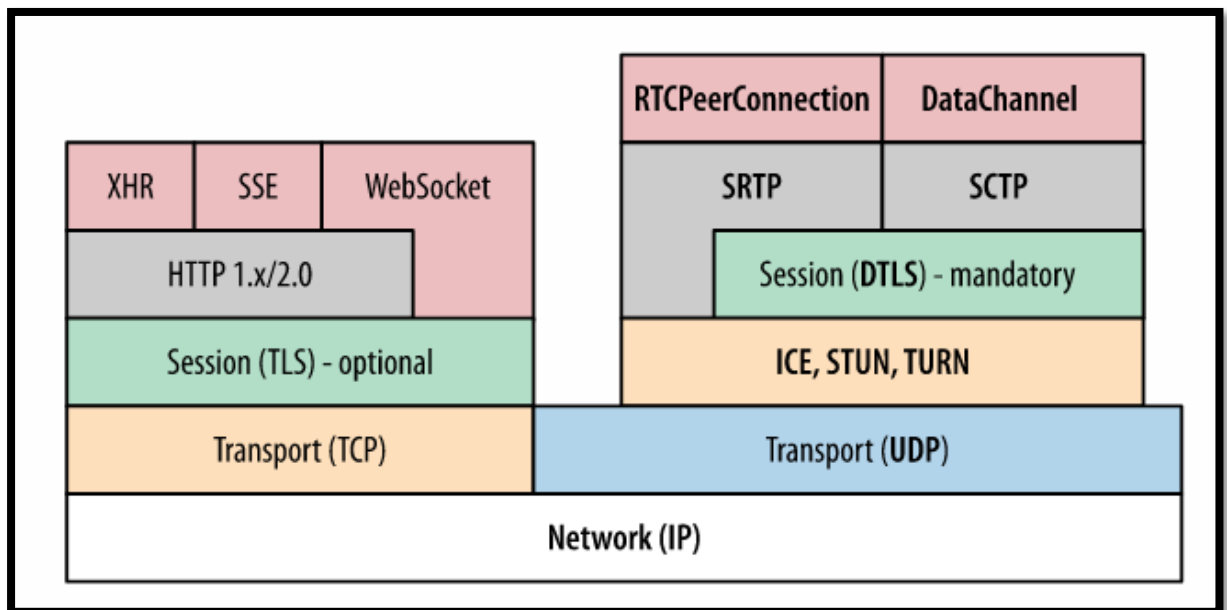
- **MediaStream**
- **RTCPeerConnection**
- **DataChannel**

#### ❖ MediaStream

- Un critère essentiel pour que le navigateur puisse offrir des fonctionnalités de téléconférence est évidemment de pouvoir accéder au contenu audio/vidéo capturé par l'appareil hôte.
- L'objet **MediaStream** permet un tel accès simplifié par la méthode **getUserMedia**, mais encapsule aussi de puissants moteurs de traitement audio/vidéo internes permettant d'en améliorer la qualité (encodage, décodage, synchronisation, gestion des fluctuations de la bande passante, dissimulation des paquets perdus, réduction des bruits dans la bande audio, amélioration de l'image dans le flux vidéo, etc.) Notons que la majorité des navigateurs vont forcer l'utilisation de HTTPS dans la page avant de permettre l'utilisation de **getUserMedia**.
- WebRTC supporte présentement différents codecs de compression, mais principalement **VP8** et **H.264** pour la vidéo et **Opus** pour l'audio.
- Une fois l'acquisition du flux multimédia enclenché, celui-ci peut être utilisé à différentes fins dans d'autres API du navigateur : **Web Audio** pour lancer la piste audio dans le navigateur, **Canvas** pour faire la capture d'une image, **CSS3** et **WebGL** pour appliquer des effets 2D/3D sur le flux ou encore **RTCPeerConnection**.

#### Transport et protocoles :

WebRTC se distingue d'abord par le choix du protocole de transport qui le soutient. En effet, WebRTC a déterminé UDP comme couche de transport (du modèle OSI) qui se prête particulièrement bien au contexte temps-réel où la latence devient critique. Si UDP n'offre aucune garantie par rapport à TCP (qui garantit l'arrivée des paquets, mais aussi l'ordre de ceux-ci), il se prête beaucoup mieux aux flux multimédias en temps réel qui peuvent généralement supporter la perte de certaines données (codecs audio et vidéo).



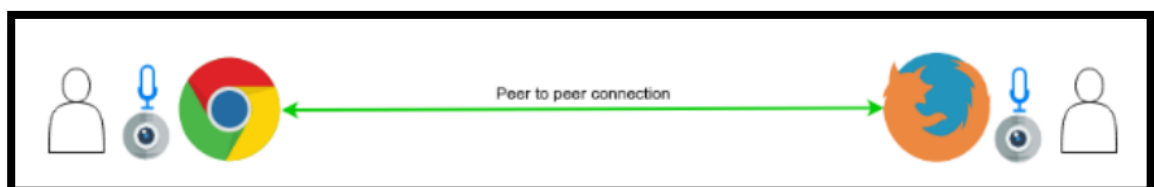
Comme le diagramme précédant en témoigne, d'autres protocoles sont mis en œuvre pour permettre notamment le cryptage du contenu, mais aussi le contrôle du débit, le contrôle de la congestion, la gestion des erreurs, etc., tous nécessaires pour éviter la submersion du réseau et la dégradation de la communication :

- **DTLS** est utilisé comme couche obligatoire de sécurité et de cryptage
- **SRTP** est le protocole utilisé pour le transport des flux audio/vidéo
- **SCTP** est utilisé comme transport des autres données applicatives

Enfin, nous expliquerons plus en détail plus tard la signification de la couche **ICE/STUN/TURN**.

## ❖ RTCPeerConnection

- **L'API RTCPeerConnection** est responsable de la communication des contenus audio et vidéo et devient donc fondamentale dans le cas qui nous intéresse. C'est par cette interface que deux pairs vont apprendre à communiquer ensemble et vont établir un tunnel de partage de contenu multimédia (par SRTP).
- **RTCPeerConnection** s'occupe des étapes d'initialisation de la connexion entre les pairs, la gestion de la session, l'envoi des flux multimédias et l'état de la communication.



*Connexion « pair à pair » de partage de contenu multimédia*

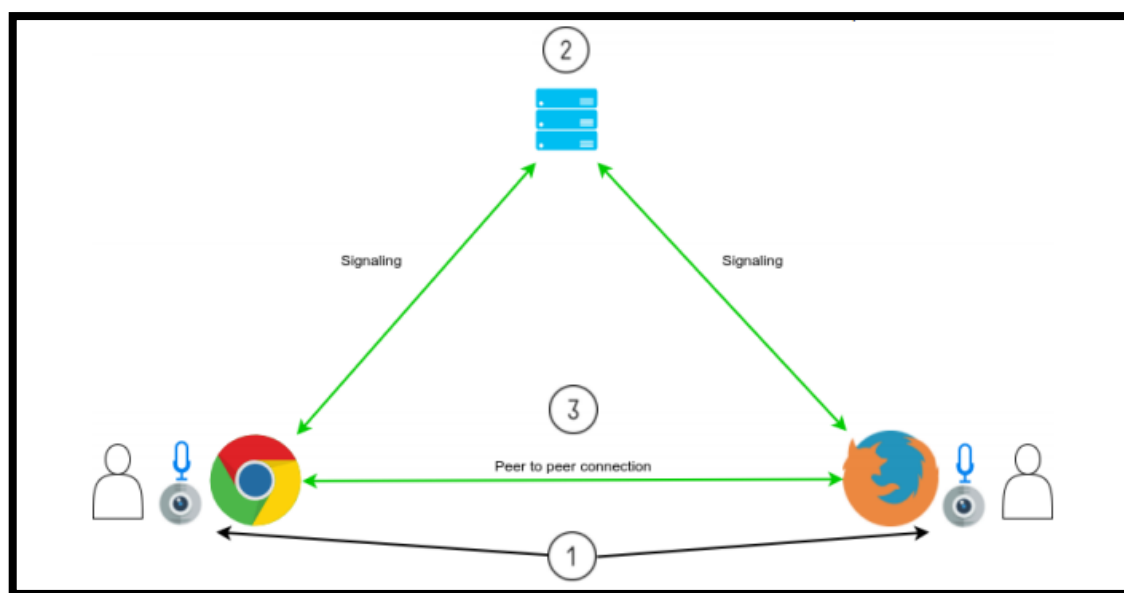
- **L'objet RTCPeerConnection** est le point d'entrée principal de l'API WebRTC. C'est ce qui nous permet d'initialiser une connexion, de nous connecter à des pairs et de joindre des informations sur le flux multimédia. Il gère la création d'une connexion UDP avec un autre utilisateur.

## ❖ DataChannel

- Cette API permet l'échange de données arbitraires (textuelles ou binaires) entre navigateurs et est entièrement dépendante de l'objet `RTCPeerConnection` en ce qu'elle nécessite une connexion déjà ouverte.
- C'est réellement l'équivalent de `WebSocket`, mais cette fois-ci entre deux pairs et à quelques différences près : la session d'échange peut être initiée par l'une ou l'autre des parties prenantes et la méthode de livraison des paquets et la fiabilité de celle-ci est configurable (au moyen de `SCTP`).

### L'initialisation de la communication :

- Il faut noter que l'initialisation de la communication (ou `signaling`) est le chaînon manquant sans lequel vos fureteurs ne pourront pas communiquer directement ensemble.
- Cette initialisation nécessaire à `RTCPeerConnection` requiert impérativement un serveur distant qui va servir d'intermédiaire aux différents participants. L'initialisation se résume à l'authentification des participants et à l'échange d'informations nécessaires à la localisation des multiples participants (adresse IP, port, `SDP` local, etc.)



*L'initialisation comme l'une des trois étapes essentielles.*

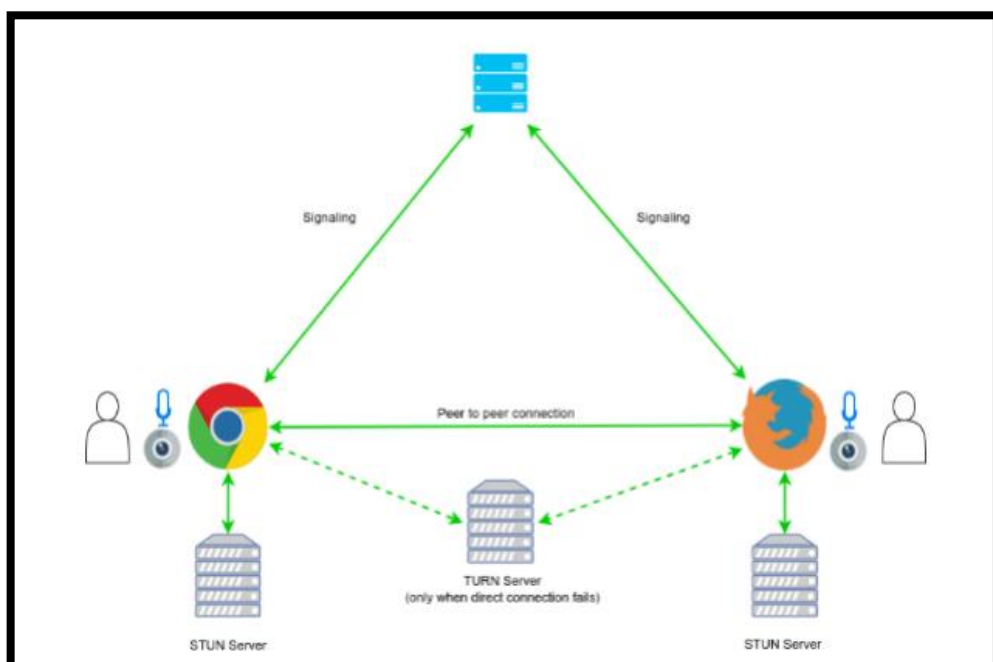
1. Le navigateur obtient l'accès au matériel d'enregistrement comme la caméra ou le microphone.
2. Chaque participant s'identifie et échange l'information nécessaire à l'établissement d'une connexion avec les autres participants.
3. Chaque participant peut maintenant partager directement son contenu multimédia avec les autres participants et la communication peut alors commencer.



## L'Internet et les serveurs ICE (STUN/TURN)

- Le réseau Internet est en fait constitué de plusieurs millions de réseaux publics et privés reliés par de nombreux câbles, routeurs et une panoplie d'autres dispositifs comme des pare-feu ou encore des routeurs NAT (Network Address Translation) ainsi que différentes restrictions déterminées par votre fournisseur de service Internet (ISP).
- La majorité des appareils disposent d'une adresse IP privée (adresse non unique d'un point de vue global) et nécessite un dispositif NAT qui s'occupe de transformer l'adresse privée en adresse publique (ainsi que les ports) pour pouvoir accéder à l'Internet. En réalité, dans la grande majorité des cas, deux fureteurs ne pourront tout simplement pas communiquer directement ensemble puisqu'aucun des deux ne connaît l'adresse IP et le port qu'il peut utiliser pour atteindre l'autre.
- C'est ici que la technique **ICE** (Interactive Connectivity Establishment) intervient pour permettre aux agents WebRTC de communiquer en faisant abstraction de la complexité des réseaux formant la réalité d'Internet. ICE va utiliser des serveurs Session Traversal Utilities for NAT (STUN) et Traversal Using Relays around NAT (TURN) pour pallier aux problèmes causés par les pare-feu, les routeurs NAT et autres restrictions potentielles.
- Un serveur **STUN** permet à l'agent de connaître les informations réseaux qu'il expose à l'externe, soit son adresse IP publique, le port, mais aussi le type de routeur NAT devant lui.
- Un serveur **TURN** est une extension à STUN lorsque celui-ci ne parvient pas à établir la connexion. Il va permettre de relayer le média d'un agent à l'autre et devra être présent durant toute la communication contrairement à STUN. TURN est une opération définitivement plus exigeante et la communication directe devrait en tout temps être privilégiée.
- En réalité, ICE fait partie intégrante de l'étape d'initialisation de la communication et voici à quoi ressemble notre nouvelle architecture après avoir ajouté nos serveurs STUN et TURN :

*Architecture  
WebRTC avec  
serveurs TURN  
et STUN.*





## **4| Sécurité**

L'un des risques de sécurité dans toute application de communication en temps réel peut augmenter lors de la transmission de données.

Finalement, le cryptage est une fonctionnalité obligatoire de WebRTC et est appliqué sur tous les composants.

WebRTC utilise deux protocoles de chiffrement normalisés :

### **Sécurité de la couche de transport en datagrammes (DTLS)**

- Un protocole standardisé intégré à un navigateur. Il est utilisé pour crypter les flux de données. Il est basé sur le protocole TLP (Transport Layer Protocol).
- Préserve la sémantique du transport car DTLS utilise le protocole UDP (User Data Protocol).
- C'est une extension de Secure Sockets Layer (SSL) ; tout protocole SSL pourrait être utilisé pour sécuriser les données WebRTC permettant un cryptage de bout en bout.

### **Protocole de transport sécurisé en temps réel (SRTP)**

- Utilisé pour crypter les flux multimédias.
- Il s'agit d'une extension du protocole de transport en temps réel (RTP), qui n'a aucun mécanisme de sécurité intégré.
- Ajoute la protection, l'intégrité et l'authentification des messages au RTP.
- Inconvénient : bien qu'il fournisse le cryptage des paquets RTP, il ne crypte pas l'en-tête.

### **Étapes pour sécuriser un lien entre 2 pairs**

1. Lance le processus de signalisation échange les métadonnées entre deux pairs.
2. Le contrôle **ICE** est effectué et **ICE** établit un canal entre les parties.
3. L'établissement de liaison **DTLS** est effectué. S'il y a des médias transportés, le **SRTP** utilise les clés qui ont été exportées à l'étape de prise de contact **DTLS**.
4. Tous les pairs ont des canaux sécurisés avec des clés qui ne sont pas connues publiquement.
5. Échangez des clés entre les pairs.

# Chapitre 3 : Création d'une Application WebRTC simple

## 1| Introduction

Dans cette partie, nous allons créer notre application WebRTC de chat Vidéo, nous avons utilisé la base de données WebRTC et Firebase Realtime pour créer cette application.

Tout d'abord, nous expliquerons comment WebRTC fonctionne pour comprendre ce que nous allons faire afin de créer notre application et de connecter les utilisateurs entre eux. Deuxièmement, nous parlerons de Firebase Database et des raisons pour lesquelles nous avons utilisé Firebase pour le serveur de signalisation de notre application et pourquoi il vaut mieux être sans serveur.

Troisièmement, nous allons configurer les modules Firebase et WebRTC et mettre en œuvre les étapes nécessaires pour créer notre première application de chat vidéo.

Et à la fin, on va faire une simulation afin de vérifier le bon fonctionnement de notre application.

## 2| Procédure de chat vidéo WebRTC

Parlons un peu de la façon dont WebRTC peut être utilisé pour configurer un chat vidéo. Supposons que nous ayons deux ordinateurs, le vôtre et celui de votre ami. Voici la procédure étape par étape nécessaire pour faire fonctionner le chat vidéo.

1. Affichez une vidéo **MediaStream** de vous-même sur votre ordinateur
2. Affichez une vidéo **MediaStream** de votre ami sur son ordinateur
3. Créez une connexion **PeerConnection** sur votre ordinateur
4. Créez une connexion **PeerConnection** sur l'ordinateur de votre ami
5. Créez une **offre** sur votre ordinateur
6. Ajoutez cette **offre** à **PeerConnection** sur votre ordinateur
7. Envoyez cette **offre** à l'ordinateur de votre ami
8. Ajoutez cette **offre** à **PeerConnection** sur l'ordinateur de votre ami
9. Générez des **candidats ICE** sur votre ordinateur
10. Envoyez ces **candidats ICE** à l'ordinateur de votre ami
11. Ajoutez des **candidats ICE** à **PeerConnection** sur l'ordinateur de votre ami
12. Créez une **réponse** sur l'ordinateur de votre ami
13. Ajoutez cette **réponse** à **PeerConnection** sur l'ordinateur de votre ami
14. Envoyez cette **réponse** à votre ordinateur
15. Ajoutez cette **réponse** à **PeerConnection** sur votre ordinateur
16. Générez des **candidats ICE** sur l'ordinateur de votre ami
17. Envoyez ces **candidats ICE** à votre ordinateur
18. Ajouter des **candidats ICE** à **PeerConnection** sur votre ordinateur
19. Affichez une vidéo **MediaStream** de votre ami sur votre ordinateur
20. Affichez une vidéo **MediaStream** de vous-même sur l'ordinateur de votre ami.

### 3| Préparation de notre environnement

#### a. Les technologies utilisées

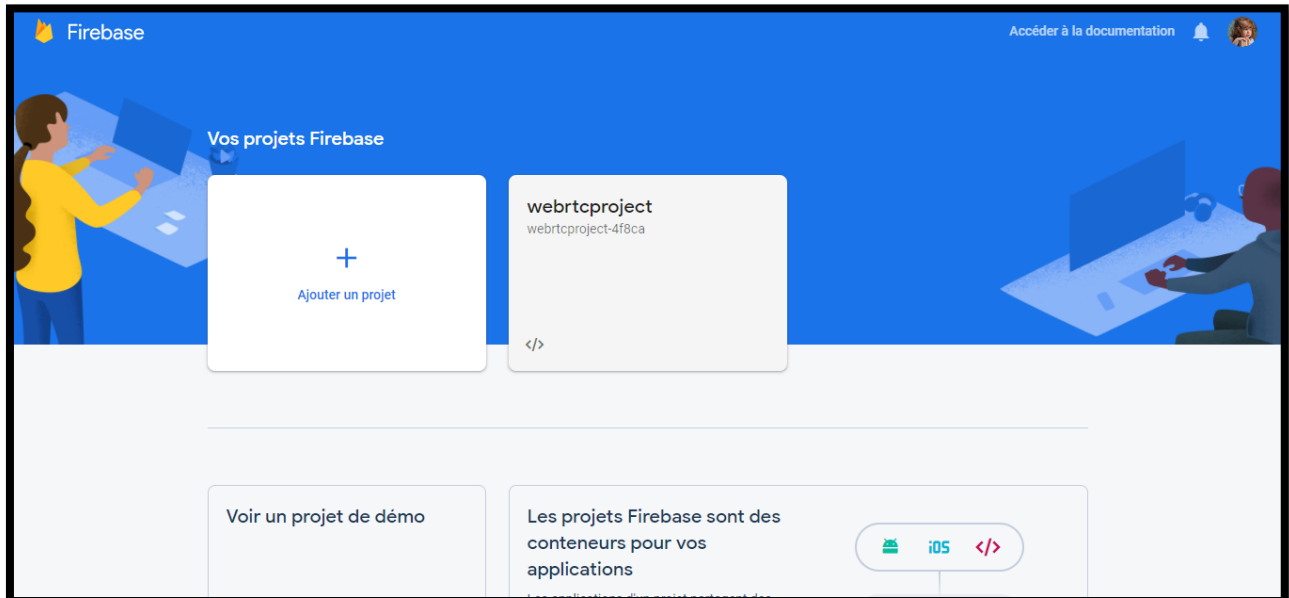
Pour que ce chat vidéo fonctionne, nous utiliserons deux technologies importantes :

- **La première technologie :** est WebRTC (Web Real-Time Communication). WebRTC est intégré à notre navigateur.
- **La deuxième technologie :** est **Google Firebase**, Firebase est une base de données en direct.
- Si vous ajoutez des données à votre base de données Firebase, un utilisateur de votre site Web n'a pas besoin d'actualiser la page pour voir les nouvelles données. Les nouvelles données apparaîtront simplement.
- Contrairement à WebRTC, vous devez importer la bibliothèque JavaScript Firebase pour pouvoir l'utiliser.
- Vous devez également créer un compte Firebase gratuit, qui permet 100 connexions simultanées. Firebase nous permettra d'envoyer et de recevoir des messages en direct, dont vous avez besoin pour que le chat vidéo fonctionne.



## b. La configuration de Firebase

1. Nous avons tout d'abord accédé à <https://firebase.google.com> et on a créé un compte gratuit. Ensuite on clique sur "Ajouter un projet".



2. Nous avons entré un nom de projet et cliqué sur « Continuer ».
  - Et on clique « Créer un projet ».



× Créer un projet(Étape 2 sur 3)

## Google Analytics for your Firebase project


Google Analytics est une solution d'analyse gratuite et illimitée qui permet le ciblage, la création de rapports et bien plus dans Firebase Crashlytics, Cloud Messaging, Remote Config, A/B Testing, Predictions, Cloud Functions et la messagerie dans l'application.





× Créer un projet(Étape 2 sur 3)


Google Analytics permet d'exploiter les fonctionnalités suivantes :


 Tests A/B ?

 Segmentation et ciblage des utilisateurs dans les produits Firebase ?

 Prédiction du comportement des utilisateurs ?

 Utilisateurs n'ayant pas subi de plantage ?

 Déclencheurs de fonctions Cloud basés sur des événements ?

 Rapports gratuits et illimités ?

☒ Activer Google Analytics pour ce projet  
Recommandation


Précédent


Continuer

× Créer un projet(Étape 3 sur 3)

## Configurer Google Analytics

Sélectionner ou créer un compte Google Analytics ?

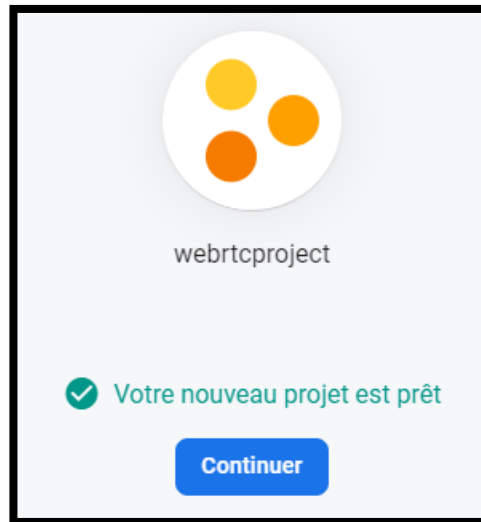
 Default Account for Firebase ▼

Créer automatiquement une propriété dans ce compte 

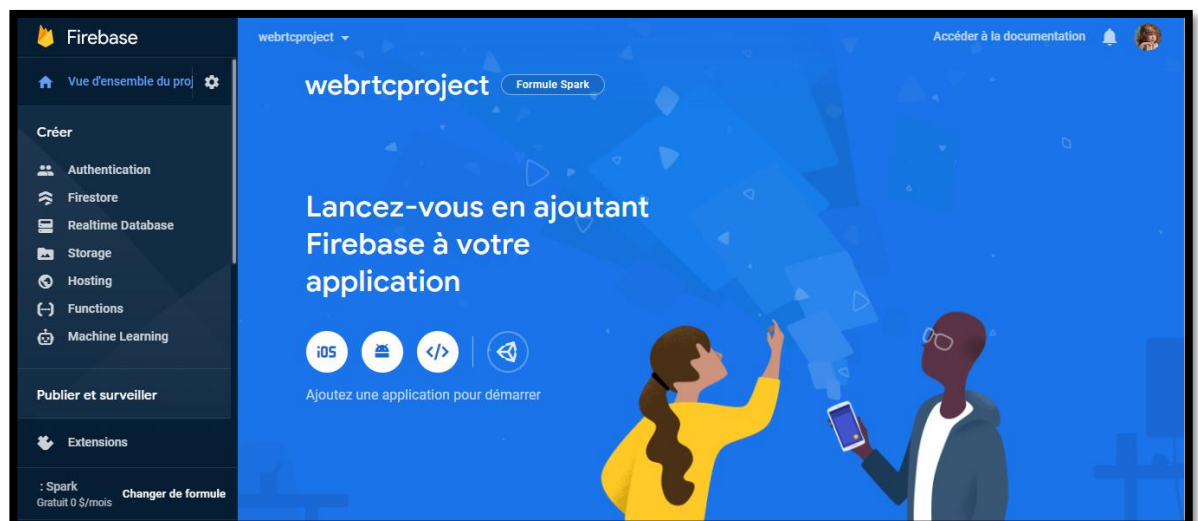
À la création du projet, une nouvelle propriété Google Analytics va être créée dans le compte Google Analytics de votre choix, puis associée à votre projet Firebase. Cette association permettra l'échange de données entre ces produits. Les données de votre propriété Google Analytics exportées vers Firebase sont soumises aux conditions d'utilisation de Firebase, et les données Firebase importées dans Google Analytics sont soumises aux conditions d'utilisation de Google Analytics. [En savoir plus](#)

Précédent

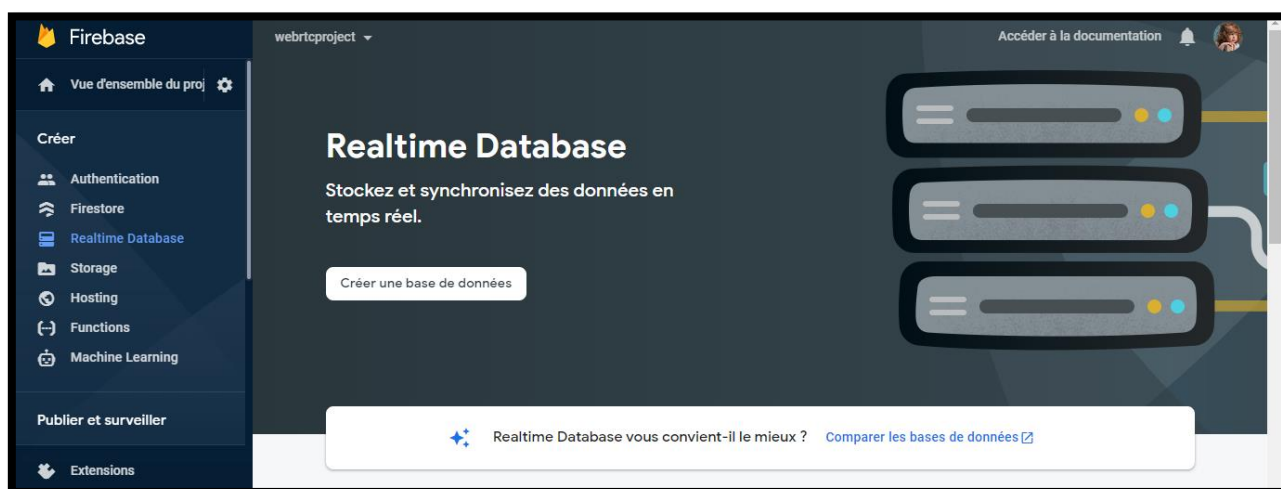
Créer un projet



3. Une fois la création du projet est terminée, nous avons cliqué sur « Ajouter Firebase à votre application Web ».



4. On a ensuite créé une base de données en temps réel.



## Configurer la base de données

1 Options de la base de données — 2 Règles de sécurité

Le paramètre d'emplacement définit l'endroit où vos données Realtime Database seront stockées.

Emplacement de Realtime Database

États-Unis (us-central1)

Annuler Suivant

- En cochant la case « Démarrer en mode test », tout le monde peut lire et écrire dans notre base de données Firebase pendant les 30 prochains jours.

## Configurer la base de données

1 Options de la base de données — 2 Règles de sécurité

Après avoir défini votre structure de données, vous devez spécifier des règles pour sécuriser vos données. [En savoir plus](#)

☐ Commencer en **mode verrouillé**  
Par défaut, vos données sont privées. L'accès client en lecture/écriture ne sera autorisé qu'en fonction de vos règles de sécurité.

☒ **Démarrer en mode test**  
Par défaut, vos données sont publiques pour permettre une configuration rapide. Toutefois, vous devez mettre vos règles de sécurité à jour dans les 30 jours pour autoriser l'accès client en lecture/écriture sur le long terme.

```
{
  "rules": {
    ".read": "now < 1622502000000", // 2021-6-1
    ".write": "now < 1622502000000", // 2021-6-1
  }
}
```

**Par défaut, les règles de sécurité en mode test autorisent tout utilisateur disposant de la référence de votre base de données à afficher, modifier et supprimer toutes les données qu'elle contient pendant les 30 prochains jours**

- Nous avons copié le code et les informations d'identification afin de les ajouter dans notre code (script.js).

Accéder à la documentation

Firestore SDK snippet

☐ Automatique ☐ CDN ☒ Configuration

Copiez et collez ces scripts en bas de votre balise <body>, et ce, avant d'utiliser les services Firebase :

```
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyCYNqWu0hIdj_b0jDcrQfkcIVU-WYbCWNs",
  authDomain: "webtrcproject-4f8ca.firebaseio.com",
  databaseURL: "https://webtrcproject-4f8ca.firebaseio.com",
  projectId: "webtrcproject-4f8ca",
  storageBucket: "webtrcproject-4f8ca.appspot.com",
  messagingSenderId: "1051352016469",
  appId: "1:1051352016469:web:d7d592b8ff2cbe3f3456dc",
  measurementId: "G-1LE2H75S0P"
};
```

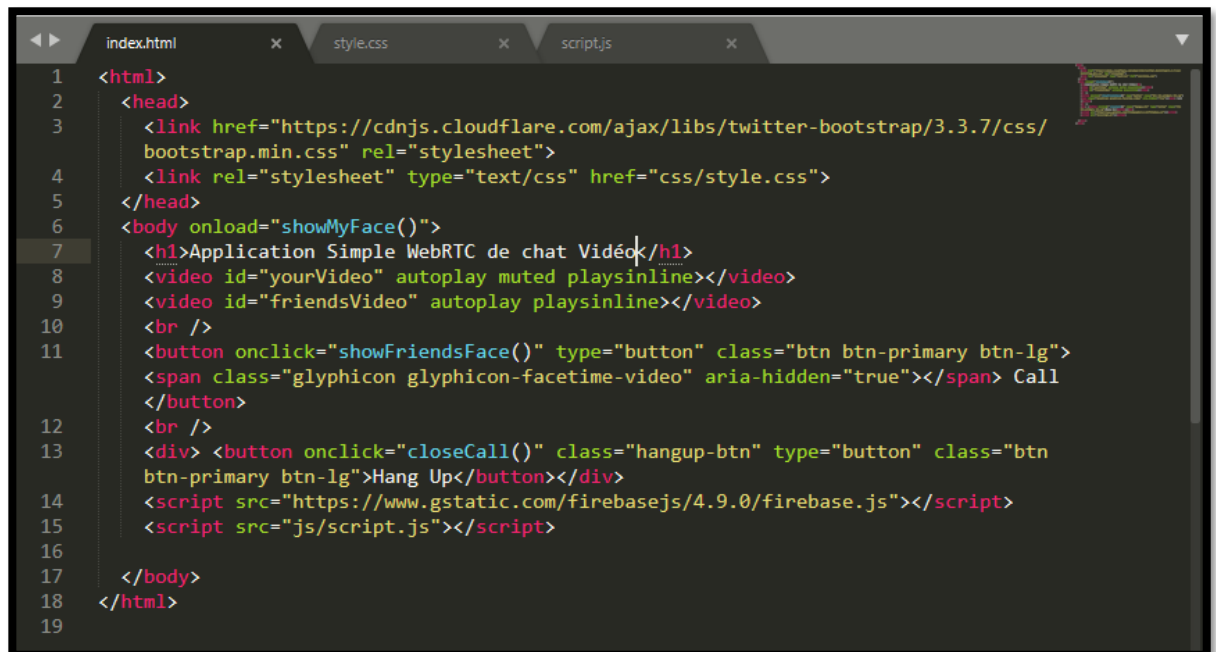
Supprimer cette application



## 4| Explication du code de notre Application

### a. Partie HTML (index.html) :

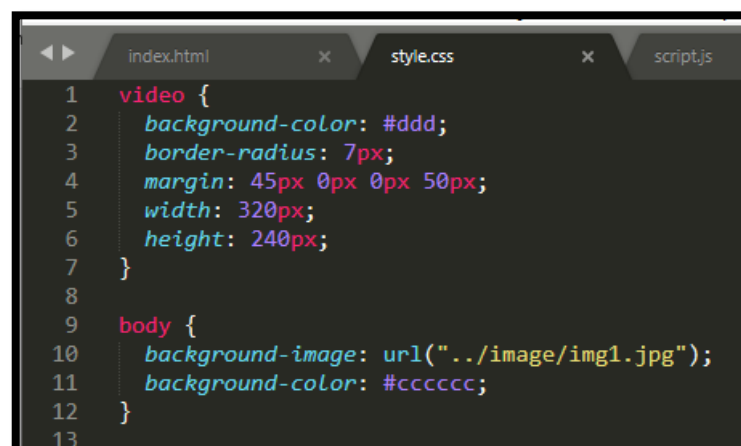
- On a commencé par charger la bibliothèque JavaScript Firebase.
- La bibliothèque CSS Bootstrap est utilisée pour rendre le bouton d'appel et le bouton de raccrochement agréable.
- Si on clique sur le bouton « **Call** », la fonction *showFriendsFace()* est appelée.
- Une fois le corps chargé, *showMyFace()* est appelé. On expliquera ces fonctions lorsque nous passerons en revue la partie JavaScript.
- Deux balises vidéo sont utilisées. L'un sera une vidéo de la personne1 et l'autre sera une vidéo de personne2.
- L'attribut de lecture automatique lance la lecture de la vidéo immédiatement après qu'on a défini la source.
- Si on clique sur le bouton « **Hang Up** », la fonction *closeCall()* est appelée.
- Passons à la partie CSS de notre projet.



```
1 <html>
2 <head>
3   <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/
4     bootstrap.min.css" rel="stylesheet">
5   <link rel="stylesheet" type="text/css" href="css/style.css">
6 </head>
7 <body onload="showMyFace()">
8   <h1>Application Simple WebRTC de chat Vidéo</h1>
9   <video id="yourVideo" autoplay muted playsinline></video>
10  <video id="friendsVideo" autoplay playsinline></video>
11  <br />
12  <button onclick="showFriendsFace()" type="button" class="btn btn-primary btn-lg">
13    <span class="glyphicon glyphicon-facetime-video" aria-hidden="true"></span> Call
14  </button>
15  <br />
16  <div> <button onclick="closeCall()" class="hangup-btn" type="button" class="btn
17    btn-primary btn-lg">Hang Up</button></div>
18  <script src="https://www.gstatic.com/firebasejs/4.9.0/firebase.js"></script>
19  <script src="js/script.js"></script>
20 </body>
21 </html>
```

### b. Partie CSS (style.css) :

- Ce code donne aux vidéos un fond gris avec des bords arrondis. Nous attribuons à chaque lecteur vidéo une largeur et une hauteur fixes en plus des marges. Idem avec le bouton. Nous fixons sa largeur et définissons ses marges.



```
1 video {
2   background-color: #ddd;
3   border-radius: 7px;
4   margin: 45px 0px 0px 50px;
5   width: 320px;
6   height: 240px;
7 }
8
9 body {
10  background-image: url("../image/img1.jpg");
11  background-color: #cccccc;
12 }
13
```



```

13
14 button {
15     margin: 6px 0px 0px 65px !important;
16     width: 654px;
17     border-radius: 8px;
18     transition-duration: 0.4s;
19 }
20
21 h1 {
22     font-family: Apple Chancery, cursive;
23     color: white;
24     margin: 45px 0px 0px 50px;
25 }
26

```

- Nous en sommes à la dernière partie du code, à savoir le JavaScript.

### c. Partie Javascript (script.js) :

- Dans cette partie nous allons expliquer juste les fonctions les plus importantes de notre code.

```

index.html x style.css x script.js x
1 //Firebase
2 var config = {
3     apiKey: "AIzaSyCYnqWu0hIdj_bQjDcrQfkIVU-WYbCwNs",
4     authDomain: "webrtcproject-4f8ca.firebaseio.com",
5     databaseURL: "https://webrtcproject-4f8ca-default-rtdb.firebaseio.com",
6     projectId: "webrtcproject-4f8ca",
7     storageBucket: "webrtcproject-4f8ca.appspot.com",
8     messagingSenderId: "1051352016469",
9     appId: "1:1051352016469:web:d7d592b8ff2cbe3f3456dc",
10    measurementId: "G-1LE2H75S0P"
11 };
12 firebase.initializeApp(config);
13

```

```

var database = firebase.database().ref();
var yourVideo = document.getElementById("yourVideo");
var friendsVideo = document.getElementById("friendsVideo");
var yourId = Math.floor(Math.random()*1000000000);

```

- **var RTCPeerConnection = window.webkitRTCPeerConnection;**  
s'assure que ce code fonctionne avec différents navigateurs. Les lignes de code suivantes nous permettent d'accéder à notre Firebase.
- *var database = firebase.database (). ref ();* → nous donne accès à la racine de notre base de données Firebase.
- *database.on ('enfant\_added', readMessage);* → fait en sorte que si vous ajoutez quelque chose à la base de données Firebase en appelant *sendMessage*, il sera automatiquement lu. En d'autres termes, chaque message inséré dans la base de données Firebase sera lu car *readMessage* est appelé chaque fois que Firebase détecte des données nouvellement insérées. Ensuite, nous définissons *yourVideo* et *friendsVideo* sur les éléments vidéo trouvés dans le HTML.

- Nous donnons à l'utilisateur un identifiant aléatoire qui nous aide à différencier les deux utilisateurs. Lorsque nous envoyons des données (objets Offre, Réponse et Candidat ICE) de notre ordinateur à l'ordinateur d'une autre personne, celui-ci doit les recevoir. Et il le fera parce que nous les enverrons via Firebase.

```
//Viagenie
var servers = {'iceServers': [{'urls': 'stun:stun.services.mozilla.com'},
{'urls': 'stun:stun.l.google.com:19302'},
{'urls': 'turn:numb.viagenie.ca','credential': 'beaver','username': 'wifkaouiyasmine@gmail.com'}]};
var pc = new RTCPeerConnection(servers);
pc.onicecandidate = (event => event.candidate?sendMessage(yourId, JSON.stringify({'ice': event.candidate})));
pc.onaddstream = (event => friendsVideo.srcObject = event.stream);
```

- Juste après avoir généré un identifiant d'utilisateur aléatoire, nous déclarons les serveurs que nous utiliserons. Nous incluons deux serveurs STUN (Google et Firefox) et un serveur TURN. Les serveurs STUN sont moins chers que les serveurs TURN, c'est pourquoi Google et Firefox permettent à quiconque d'accéder gratuitement à leurs serveurs STUN. Les serveurs TURN sont plus difficiles à trouver gratuitement, mais ils existent.
- Passons à la fonction *showMyFace()*.

```
function showMyFace() {
  navigator.mediaDevices.getUserMedia({audio:true, video:true})
    .then(stream => yourVideo.srcObject = stream)
    .then(stream => pc.addStream(stream));
}
```

- Lorsque nous appelons *getUserMedia*, le navigateur demande l'autorisation d'accéder à notre caméra. Cela renverra un objet *MediaStream*, sur lequel l'ensemble peut définir *yourVideo.srcObject*.
- Ces deux lignes montrent une vidéo de *personnel* sur son ordinateur. Ajoutez ensuite ce même objet *MediaStream* à l'objet *PeerConnection* de la *personnel*. La *Personne2* doit faire de même. Cette fonction est appelée dès le chargement de la page, donc *personnel* verra son visage une fois qu'elle chargera la page.
- Une fois que *personnel* et *personne2* ont ouvert la page d'index, l'un d'eux doit appuyer sur le bouton d'appel. Cela appellera la fonction *showFriendsFace()*.

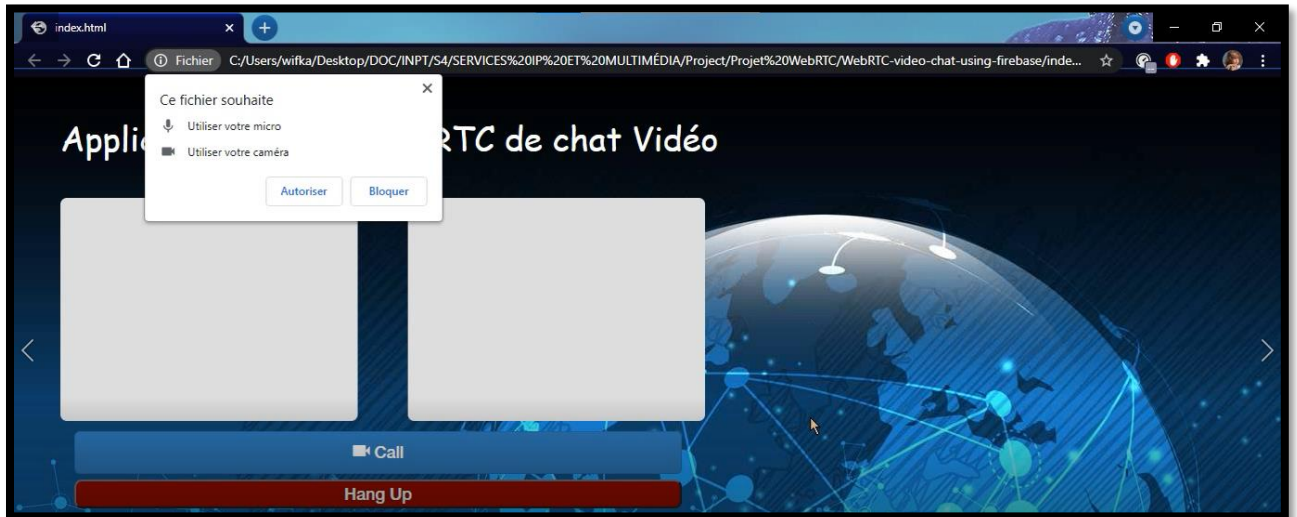
```
function showFriendsFace() {
  pc.createOffer()
    .then(offer => pc.setLocalDescription(offer) )
    .then(() => sendMessage(yourId, JSON.stringify({'sdp': pc.localDescription})) );
  showConnectedContent();
}
```

- Nous créons un objet *Offer* en appelant *pc.createOffer()*. Cela renverra un objet *Offer*.
- Nous définissons notre description locale pour cette offre en appelant *pc.setLocalDescription(offer)*. Enfin, nous envoyons cet objet *Offer* à notre ami en appelant *sendMessage*.

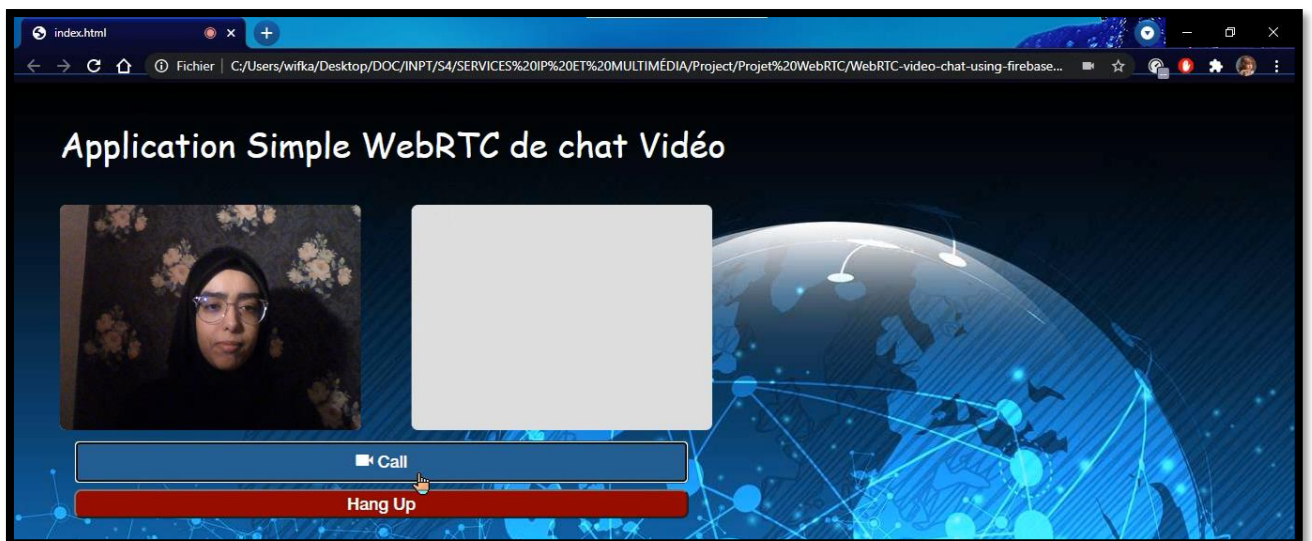
## 5| Test

Afin de vérifier la réussite de notre application WebRTC de chat vidéo, on a simulé un appel vidéo.

- J'ai tout d'abord autorisé à mon navigateur d'utiliser le micro et caméra de mon PC.



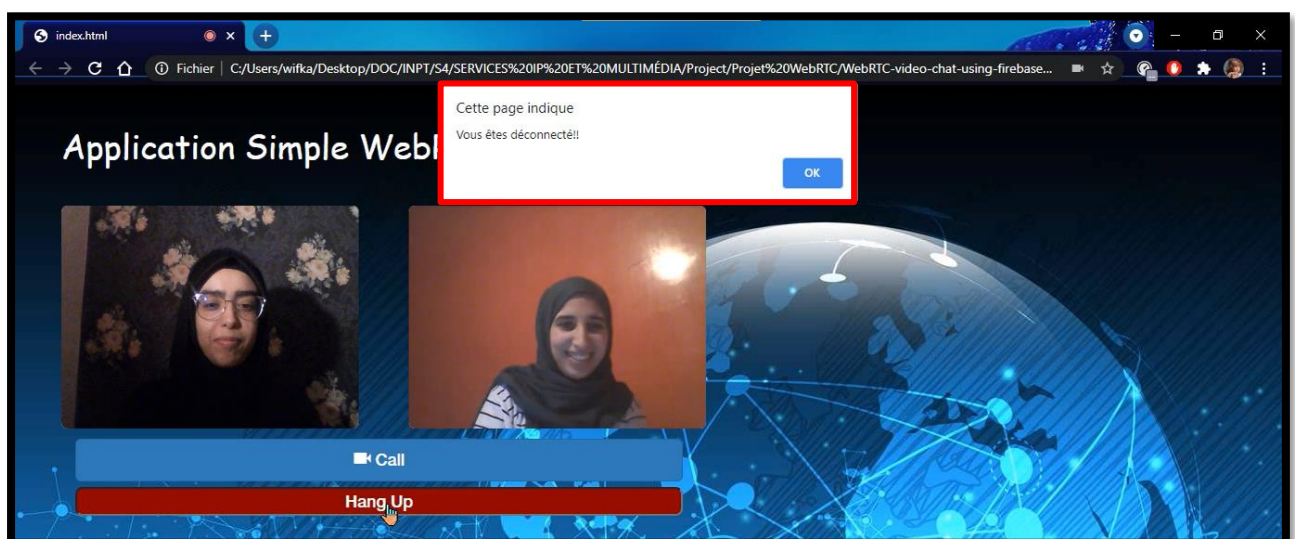
- Une fois la caméra est activée, j'ai cliqué sur Call afin d'appeler Assia Taifi.



- L'appel est réussi, chacun de nous arrive à voir l'autre et entend sa voix aussi.



- Ensuite j'ai cliqué sur Hang up afin de finir l'appel.
- Un message s'affiche indiquant que j'ai été déconnecté.



## 6| Conclusion

Dans cette partie, nous avons fait la création d'un chat vidéo en utilisant WebRTC en utilisant HTML, CSS et Javascript. Aucun plug-in ou bibliothèque n'est requis pour cette application (à part Firebase et Bootstrap). Bootstrap est utilisé pour rendre la démo plus jolie. Et Firebase est utilisé pour envoyer et recevoir des objets. C'est ce qu'on appelle un serveur de signalisation. Vous n'avez pas besoin d'utiliser Firebase comme serveur de signalisation. Vous pouvez utiliser Socket.io au lieu de Firebase comme serveur de signalisation.



## Conclusion générale

Il ne fait désormais plus aucun doute que La technologie WebRTC révolutionne les communications internationales, moyennant l'usage de navigateur Web, sans besoin d'aucune installation.

Ce projet nous a donné l'opportunité d'explorer cette technologie tout en touchant ces différentes parties théorique, technique et pratique.

On a pu faire une étude générale du WebRTC et on a réussi à créer notre propre application qui nous a permis de pratiquer tous ce qu'on a acquis durant notre recherche.

Il y'a aussi toutes les démarches qui ne sont pas visibles et qui rendent enrichissant un tel travail : s'organiser sur les plans personnels et collectifs, écouter et respecter l'opinion des autres, savoir communiquer malgré ses conditions particulières de la pandémie.

Bref, malgré qu'on n'ait pas pu ajouter plein d'autres idées et options dans notre application, à cause de la contrainte du temps, on a pu renforcer nos compétences et nos connaissances.

# Références

## ❖ Livres utilisés :

- Ristic, D. (2015). *Learning WebRTC*. Packt.
- Romano, S. L. (2014). *Real-Time Communication with WebRTC*. O'REILLY.
- Sergiienko, A. (2015). *WebRTC Cookbook*. Packt.

## ❖ Quelques sites utilisés :

- <http://www.communication-blog.com/post/2017/01/23/le-webrtc-la-revolution-de-la-communication-en-temps-reel>
- <https://www.programmersought.com/article/59643918745/>
- <https://fr.slideshare.net/BassirouDime1/mise-en-place-dun-systme-de-classes-virtuelles-utilisant-le-webrtc-openfireldapphtmlcssjavascript>
- <https://www.html5rocks.com/fr/tutorials/webrtc/basics/>
- <http://dspace.univ-tlemcen.dz/bitstream/112/10056/1/Realisation-dun-systeme-collaboratif-pour-StudyPress.pdf>
- <https://hal.archives-ouvertes.fr/hal-01339340/document>