1. Avvio del progetto

1.1. Introduzione

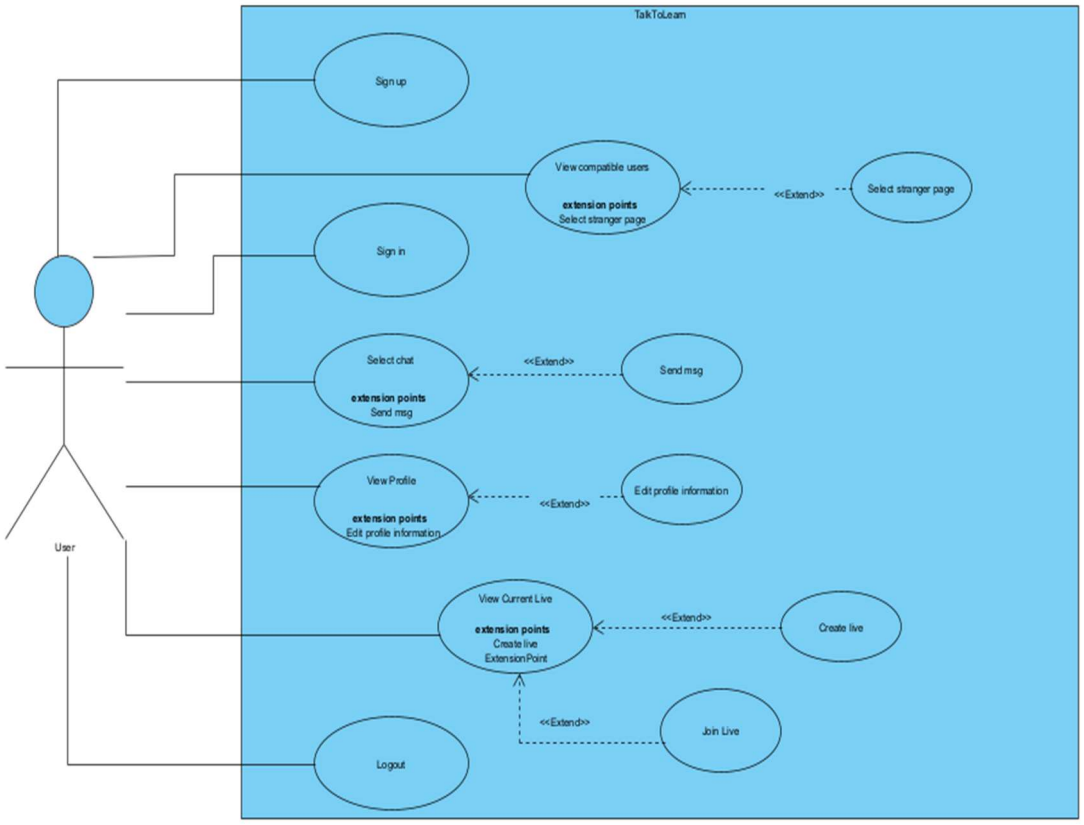
Lo scopo è quello di realizzare un'applicazione web che permetta agli utenti di insegnare e migliorare diverse lingue contemporaneamente. L'obiettivo di Talk-To-Learn è quello di rendere il viaggio di apprendimento o miglioramento di una nuova lingua molto più divertente, naturale e vivace, facendo dimenticare alle persone le noiose lezioni sui libri. In effetti, avere una vera interazione è il modo migliore per imparare una nuova lingua. Attraverso l'applicazione, l'utente può:

- *Chatta con persone provenienti da tutto il mondo*
- *Migliora la sua pronuncia*, grazie alla possibilità di accedere ai live streaming che altri utenti stanno facendo.

All'utente verrà chiesto, al momento della registrazione, di definire la lingua che vuole imparare e quella che conosce. In questo modo, appariranno tutti gli utenti complementari con cui può iniziare conversazioni.

2. Specifica dei requisiti

2.1. Diagramma dei casi d'uso



2.2.Descrizione dei casi d’uso

- **Registrazione:** l'utente richiede al sistema di registrarsi fornendo prima la sua email e password ad Auth0. Deve quindi selezionare due lingue e altre informazioni come hobby, lavoro, data di nascita ecc., al fine di creare una pagina del profilo.

- **Accedi:** l'utente accede tramite Auth0 fornendo i propri dati di accesso.

- **Visualizza utenti compatibili:** l'utente richiede al sistema di visualizzare tutti i suoi utenti complementari.

- **Seleziona pagina sconosciuta:** l'utente seleziona uno degli utenti complementari per visualizzare la pagina del suo profilo.

- **Seleziona chat:** l'utente seleziona una delle chat che ha avuto con altri utenti per inviare un messaggio.

- **Visualizza profilo:** l'utente richiede al sistema di visualizzare la pagina del proprio profilo.

- **Modifica informazioni profilo:** l'utente apporta una modifica alle informazioni fornite durante la fase di registrazione.

- **Visualizza corrente in tempo reale:** l'utente richiede al sistema di visualizzare tutte le vite eseguite da altri utenti.

- **Partecipa in diretta:** l'utente seleziona una delle trasmissioni in diretta per parteciparvi .

- **Crea live:** l'utente crea una trasmissione in diretta fornendo nome e descrizione.

- **Logout:** l'utente si disconnette dall'applicazione.

2.3. Requisiti non funzionali

- **Usabilità:** l'interfaccia deve essere intuitiva e facile da usare per l'utente.

- **Prestazioni:** la velocità di caricamento della pagina e la reattività dell'applicazione devono essere elevate.

- **Scalabilità**: l'applicazione deve essere in grado di gestire una crescita del carico di lavoro senza compromettere le prestazioni.

- **Affidabilità:** l'applicazione deve essere disponibile e funzionante senza interruzioni.

- **Sicurezza:** l'applicazione deve proteggere i dati sensibili degli utenti e prevenire gli attacchi informatici.

- **Compatibilità:** l'applicazione deve funzionare correttamente su diversi browser.

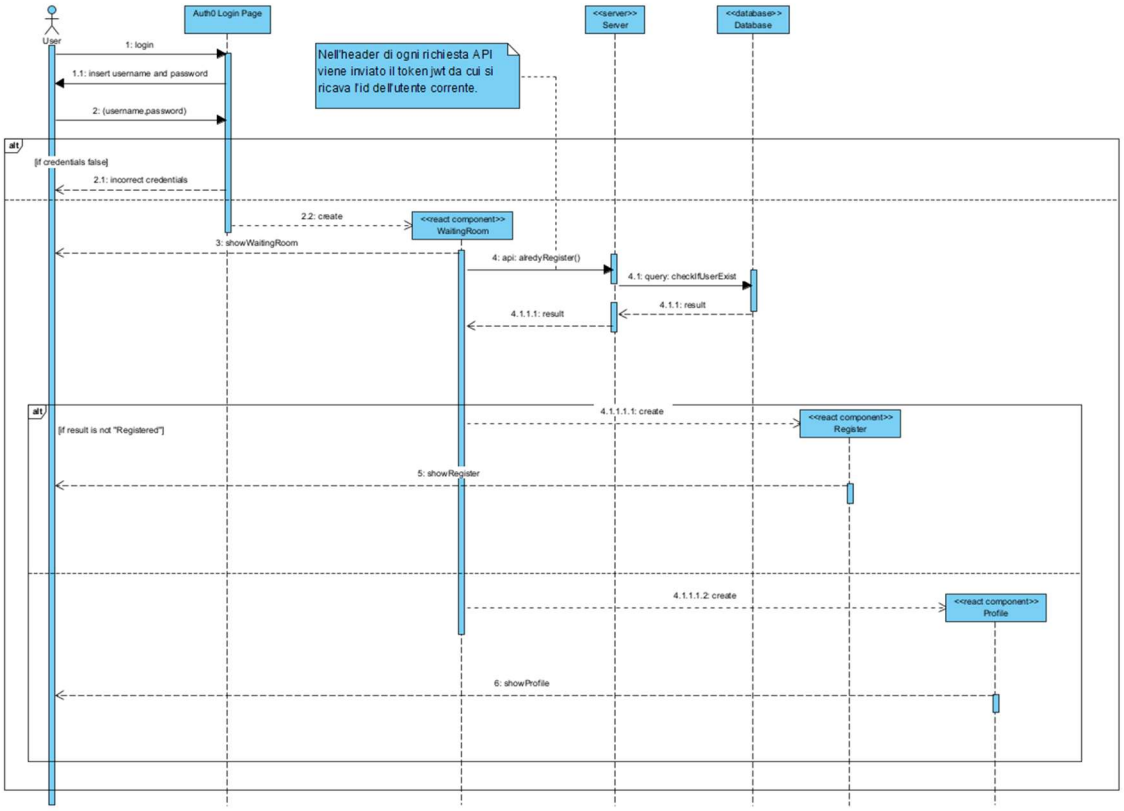
Sequence diagrams

Nei diagrammi seguenti vengono descritti i casi d'uso più importanti e viene mostrato il flusso di interazioni necessarie per eseguire l'azione.

I casi d'uso descritti sono:

- Sign In
- Sign Up
- Select chat
- View compatible users
- Join live

Sign In

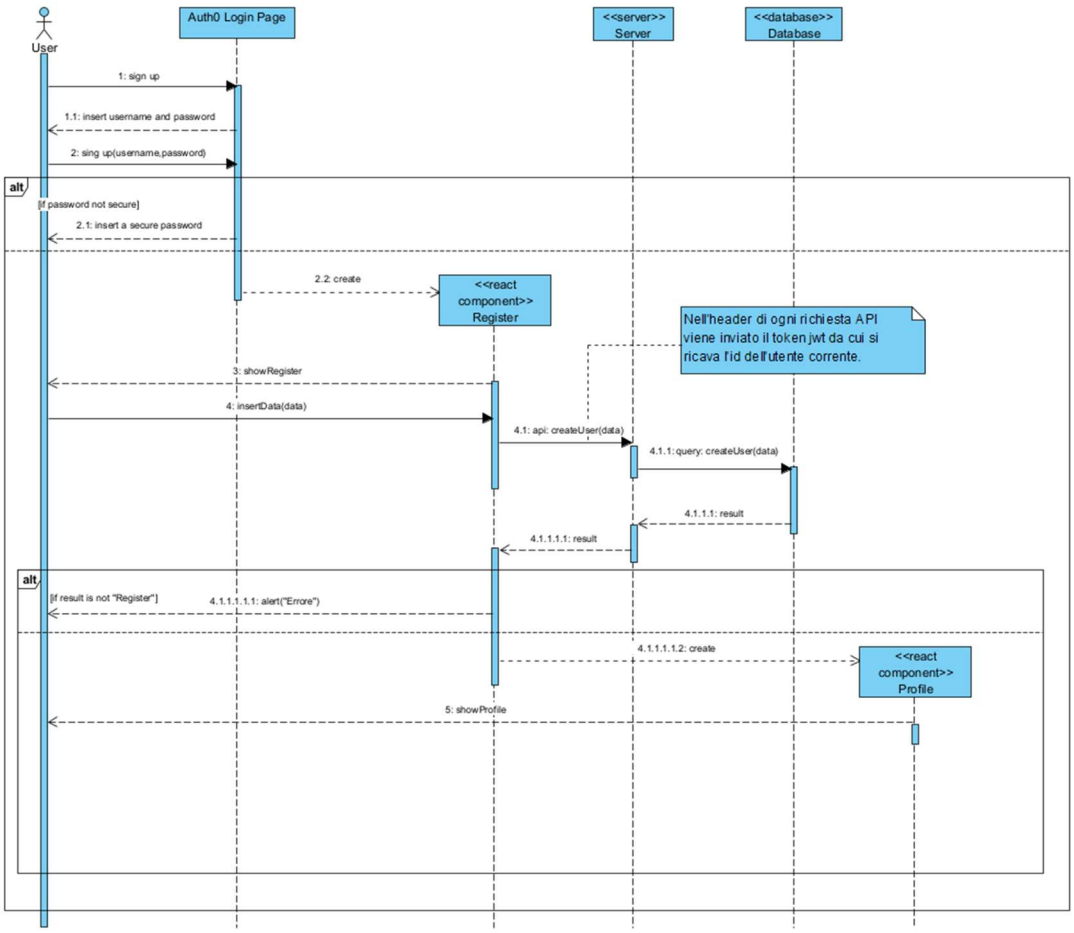


Il seguente diagramma di sequenza descrive il flusso di interazioni per il login.
Quando l'utente clicca sul pulsante di login viene contattato il server Auth0 responsabile dell'autenticazione; all'utente viene quindi mostrato un modulo in cui inserire le credenziali di accesso su cui Auth0 esegue un controllo di validazione.
Se l'accesso mediante Auth0 ha successo, viene creato e mostrato il componente *WaitingRoom*, in cui si verifica se l'utente ha già completato la registrazione o meno; in base al risultato di questa verifica si possono avere due scenari:

1. **L'utente ha già completato la registrazione:** il componente *Profilo* viene creato e mostrato all'utente con tutti i suoi dati.
2. **L'utente non ha ancora completato la registrazione:** viene creato e visualizzato il componente *Registra*, contenente i campi di registrazione richiesti (nome, lingua madre, avatar, ecc.).

Sign Up

Il seguente diagramma di sequenza descrive il flusso di interazioni per effettuare la registrazione.



Quando l'utente fa clic sul pulsante di registrazione viene contattato il server Auth0 responsabile della registrazione; all'utente viene quindi mostrato un modulo in cui inserire le

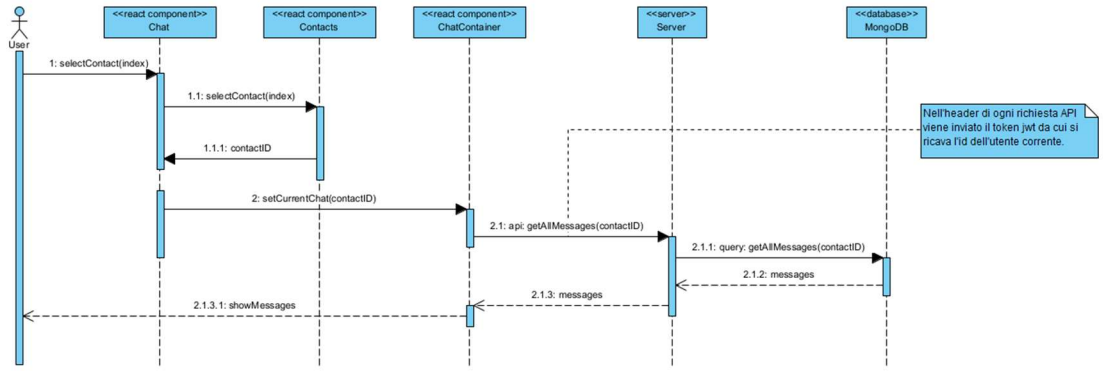
credenziali con cui desidera registrarsi e su cui Auth0 esegue un controllo di validità, in particolare viene eseguito anche un controllo sulla difficoltà della password.

Se la registrazione tramite Auth0 va a buon fine, viene creato e mostrato il componente *Register*, in cui l'utente può completare la registrazione compilando i campi richiesti (nome, lingua madre, avatar, ecc.). Questi dati saranno inviati al server tramite una richiesta POST; il server sarà responsabile della creazione dell'utente nel database MongoDB.

Se le operazioni hanno successo, il componente Profilo viene creato e visualizzato all'utente con i dati inseriti durante la registrazione, altrimenti viene restituito un messaggio di errore.

Select chat

Il seguente diagramma di sequenza descrive il flusso di interazioni innescato dalla scelta di una chat.

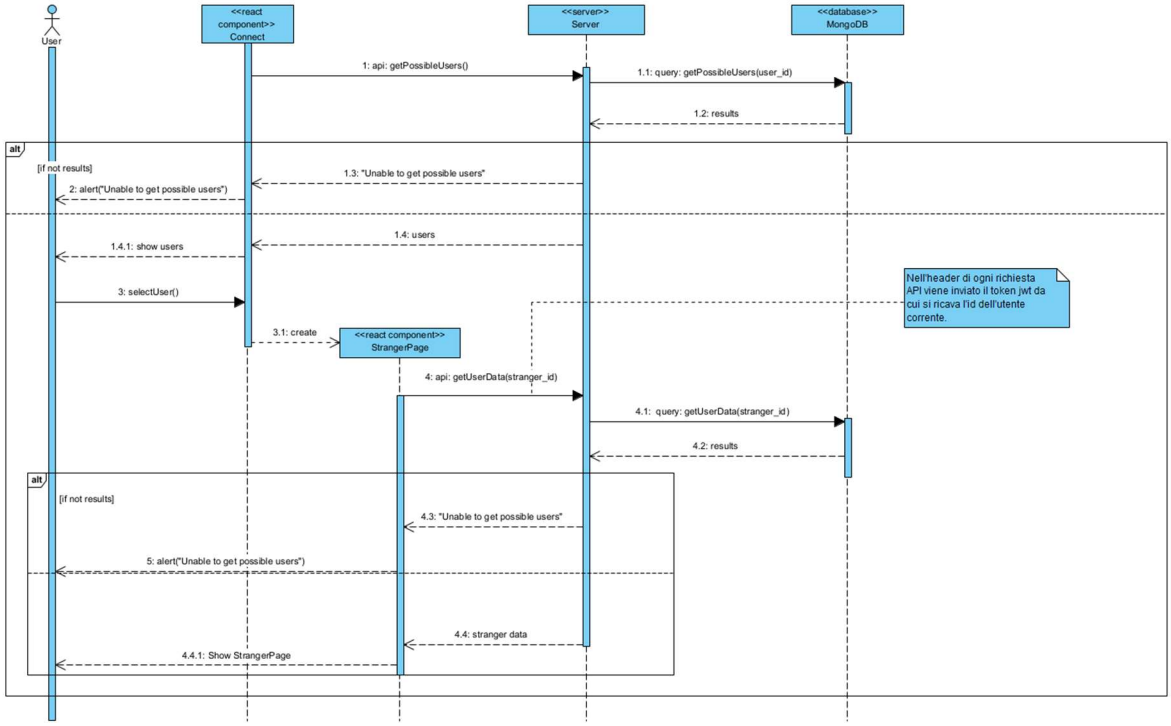


Quando l'utente clicca sulla chat desiderata, vengono coinvolti i componenti React: *Chat, Contacts e ChatContainer*.
Il componente padre Chat invia l'indice della chat scelta dall'elenco al componente figlio *Contacts*, che a sua volta restituisce l'id del contatto a cui la chat è associata. Questo id viene utilizzato da ChatContainer per visualizzare la chat corrispondente; prima di ciò, però, viene effettuata una richiesta GET al server per ottenere i vecchi messaggi scambiati con l'utente; il server effettua quindi una query al database per ottenere questi messaggi.

View compatible users

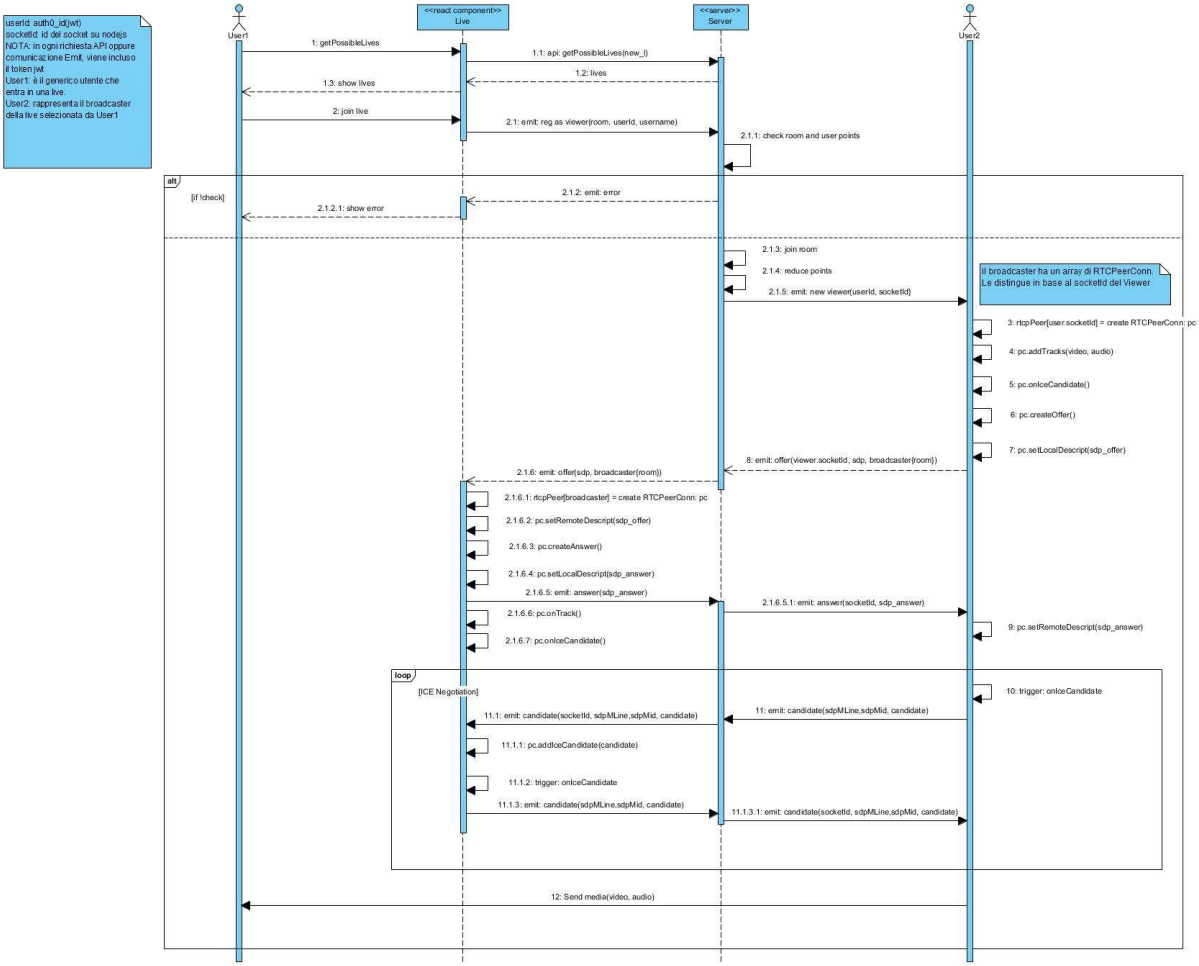
Il seguente diagramma di sequenza descrive il flusso di interazioni necessarie per visualizzare gli utenti compatibili all'utente dell'applicazione.
Quando il componente *Connect* viene caricato, viene effettuata una richiesta GET al server per ottenere tutti gli utenti compatibili; il server esegue quindi una query al database MongoDB per ottenere gli utenti.

Se non esistono utenti compatibili, viene restituito un messaggio di errore, altrimenti gli utenti vengono visualizzati. L'utente ha anche la possibilità di visualizzare il profilo di un utente compatibile. Facendo clic su di esso, viene creato il componente *StrangerPage*, che effettua



una richiesta GET al server utilizzando come parametro l'id dell'utente selezionato per ottenerne i dati. Se il processo ha successo, il profilo viene visualizzato, altrimenti viene visualizzato un messaggio di errore.

Join live



È riportato in figura il sequence diagram della funzionalità “Join live”, grazie alla quale un utente può collegarsi come spettatore ad una live già esistente, si assume che l’utente2, ovvero il Broacaster, sia già collegato al canale di signalling ed abbia istanziato la live. Con il termine “Viewer” ci riferiamo all’utente1, cioè l’utente che desidera collegarsi alla diretta.

1. Il Viewer richiede tramite API al server la lista delle live attive

2. Il Viewer selezionata una live a cui vuole partecipare, si collega al canale si signalling tramite Socket.IO ed invia l’evento di “reg as viewer” specificando il proprio ID (Auth0) ed il codice della stanza a cui desidera collegarsi “room”.

3. Il Server riceve e controlla che l’utente1 abbia abbastanza credito (10 punti) e che il codice “room” corrisponda ad una live attiva.

    a. In caso di esito NEGATIVO, viene restituito un messaggio di errore al Viewer
    b. In caso di esito POSITIVO, viene permesso all’utente1 di unirsi alla stanza e il credito ad egli associato viene decrementato. Infine, il Server manda al Broadcaster, associato alla live selezionata, l’evento “new viewer”

    specificando l’ID (Auth0) e il sockeId del Viewer, cioè l’id che identifica nel Server la connessione con il Viewer; quest’ultimo, inoltre, verrà utilizzato durante la negoziazione SDP e ICECandidate per specificare a quale socket è indirizzato un messaggio. Si prosegue al punto 4.

4. Il Broadcaster dopo aver ricevuto l’evento di “new viewer” da inizio alla negoziazione SDP e ICECandidate:

    a. Viene creata una “PeerConnection” ed aggiunta al dizionario rtcpPeer, il quale serve per memorizzare le connessioni dei vari Viewer al Broadcaster, in quanto si tratta di uno streaming One-To-Many. La chiave utilizzata per il dizionario rtcpPeer è l’id della Socket relativo al Viewer. Infine viene eseguito *“.addTracks()”* per aggiungere gli stream audio e video del Broadcaster alla connessione appena creata.
    b. Viene registrato l’evento “*.onIceCandidate()”*, esso viene chiamato ogni volta che il Layer ICE trova un nuovo candidate, la callback invierà il candidate al Viewer tramite il canale di signalling.
    c. Viene creata l’offerta SDP tramite “*.createOffer()*”, cioè viene generata un’offerta SDP(sdp_offer) dal Broadcaster, la quale contiene informazioni relative agli stream audio e video aggiunti alla PeerConnection, sarà successivamente inviata al Viewer.
    d. Il Broadcaster esegue *“.setLocalDescriptor(sdp_offer*)” quindi memorizza l’offerta appena generata come configurazione locale. Infine, l’offerta viene inviata tramite signalling al Viewer con l’evento “*offer*”, in particolare viene inviato l’sdp_offer e viene specificato il socketId del Viewer a cui si vuole inviare *l’offer*, quest’ultimo sarà usato poi da Node per capire a chi inoltrare l’evento. Dato che è stata eseguita la funzione *“.setLocalDescrp”* il Layer ICE

relativo al Broadcaster inizierà a calcolare ICECandidate quindi triggerare l'evento di .onIceCandidate.

5. L'evento "*offer*" contente *sdp_offer* viene inoltrato dal Server al Viewer, il quale eseguirà i seguenti passaggi:

    a. Viene creata una "*PeerConnection*"
    b. Viene eseguito *".setRemoteDescription(sdp_offer)",* cosi da memorizzare la configurazione SDP del peer remoto(Broadcaster)
    c. Il Viewer genera una answer SDP tramite "*.createAnswer()*", dove genera un SDP(sdp_answr) di risposta con la propria configurazione. Si nota che non è stato eseguito nessun evento di *".addTrack"* dato che il Viewer non manda nessun flusso audio\video al Broadcaster.
    d. Il Viewer esegue *".setLocalDescription(sdp_answr)"*. Dato che è stata eseguita la funziona.setLocalDescrp il Layer ICE relativo al Broadcaster inizierà a calcolare ICECandidate quindi a far scattare l'evento di *".onIceCandidate".*
    e. Viene inoltrata l'evento di "answer" al Server, diretto al Broadcaster. Il Viewer inserisce nell'evento (sdp_answr), il Server aggiunge all'evento anche l'id socketId del Viewer e lo inoltra al Broadcaster. Quest'ultimo una volta ricevuto l'evento eseguirà la funzione *".setRemoteDescription(sdp_answr)"* sulla PeerConnection relativa al Viewer che ha inviato l'answer
    f. Il Viewer registra l'evento "*.onTrack()*", il quale verrà eseguito nel momento in cui lo stream audio\video del Broadcaster sarà disponibile, la callback non farà altro che collegare lo stream al componente del DOM <video>.
    g. Infine, sarà registrato l'evento *".onIceCandidate()"*, esso viene chiamato ogni volta che il Layer ICE trova un nuovo candidate, la callback invierà il candidate al Broadcaster tramite il canale di signalling.

Alcuni step non avvengono come descritto nel diagramma, ad esempio la fase di ICE Candidate negotiation inizia non appena un peer esegue .setLocalDescription(). Ogni qual volta un peer calcola un ICE Candidate, lo invia all'altro peer tramite signalling. Una volta trovati candidate compatibili inizia la trasmissione audio\video, anche se lo scambio di ICE Candidate continua anche durante la trasmissione.

Create live

E' riportato in figura il sequence diagram della funzionalità "Create live", grazie alla quale un utente può creare una diretta assumendo il ruolo di "Broadcaster".

1. Il Broadcaster esegue .getUserMedia() per ottenere l'accesso al ai media device locali, viene mostrato all'utente un prompt nel quale può selezionare quali media utilizzare. Se la procedure va a buon fine viene richiamata la funzione .then() la quale associa lo stream ottenuto alla sorgente dell'oggetto video e ne permette poi la visualizzazione.
2. A questo punto, il Broadcaster invia tramite socket l'evento "registe as broadcaster" al Server, specificando l'id della diretta che vuole istanziare (viene usato l'id Auth0 dell'utente), il nome della stanza, la descrizione e la lingua che verrà utilizzata nella stanza (native_l).

3. Il Server ricevuto l'evento controlla che tutti i parametri siano validi e in caso positivo, memorizza le informazioni della diretta appena creata. Inoltre rende visibile la diretta agli utenti compatibili che richiedono la lista di dirette attive, cosi consentendo l'ingresso come Viewer.