Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Collaborazione real-time in una piattaforma Web di sviluppo software

Tesi di Laurea in Tecnologie Web/Internet

Relatore:
Chiar.mo Prof.
Angelo Di Iorio

Presentata da:
Antonio Marino

Sessione III
Anno Accademico 2013/2014

Indice

Capitolo 1

Introduzione

La disponibilità di connessioni Internet affidabili e veloci non solo in casa o in ufficio, ma anche in molti luoghi pubblici come piazze, aeroporti e negozi, ha cambiato il nostro modo di vivere. Attraverso dispositivi mobili come piccoli computer, tablet e smartphone, siamo in grado di navigare il Web e accedere ad una vasta gamma di servizi on-line. Grazie alla disponibilità di quest'infrastruttura il Web è in grado di offrire tantissimi servizi e tool dedicati al business o alle attività quotidiane. In questo contesto il cloud computing [1] si sta affermando come paradigma guida per la distribuzione di infrastrutture, piattaforme e risorse in rete, evitando la necessità di possedere hardware costoso e ingombrante o complessi sistemi software. Inoltre, è sempre più diffuso il concetto di servizio cloud come paradigma per l'offerta di servizi Web-based basati su un'architettura aperta e distribuita grazie alla quale è possibile condividere informazioni e risorse e che consente la collaborazione on-line [2]. La collaborazione basata sull'utilizzo del Web e sui servizi cloud si può presentare sotto forme diverse. Per questo motivo è un argomento che riguarda molte discipline. Settori come l'educazione, la medicina e la scienza in generale, il commercio on-line o i videogiochi, solo per citarne alcuni, sono tutti contesti nei quali può essere molto utile e spesso indispensabile l'utilizzo di una qualche forma di collaborazione. Basti pensare per esempio ai servizi di e-learning, dove la collaborazione è necessaria per

permettere l'interazione tra un docente e un gruppo di studenti, o ai servizi di e-shopping che prevedono la comunicazione tra acquirente e venditore o ancora, in ambito scientifico, alla collaborazione tra ricercatori nella stesura di un articolo o tra programmatori nello sviluppo di un software. Tutte le attività appena citate, anche se si riferiscono a contesti totalmente indipendenti e separati, hanno un fattore comune: la condivisione, attraverso il Web, di una risorsa in tempo reale tra più utilizzatori. La condivisione può riguardare ad esempio un video in cui vengono catturate le immagini del volto e la voce di chi partecipa ad una attività collaborativa, un documento digitale condiviso tra i collaboratori oppure un oggetto più complesso come può essere un ambiente di sviluppo software.
Per permettere la condivisione delle risorse e il loro utilizzo come oggetto di una collaborazione in tempo reale esistono diverse soluzioni offerte sul Web. Il tipo di collaborazione più diffuso è sicuramente la videochat, in quanto ha un range di utilizzatori molto ampio e vario. Grazie a strumenti come Skype[1] la videochat viene utilizzata per collaborazioni di ogni tipo, sia in ambiti professionali e lavorativi che personali. Recentemente, grazie agli sviluppi nelle tecnologie Web, si è aperta una nuova frontiera nell'ambito della collaborazione audio/video: la possibilità di utilizzare il browser come strumento per videochiamare, senza il bisogno di utilizzare software da installare sul proprio computer. Questa nuova tecnologia è chiamata WebRTC.
Tra i sistemi di collaborazione più comuni ci sono anche i cosiddetti editor collaborativi, tra i quali uno dei più famosi è Google Docs[2]: si tratta di strumenti che permettono ad una moltitudine di utenti di editare simultaneamente lo stesso documento di testo. Essi possono essere più o meno avanzati ed essere dotati di funzionalità aggiuntive come la chat, la distinzione degli utenti operanti sul documento attraverso forme o colori specifici per ogni utente, il resoconto delle modifiche apportate al documento da ciascun collaboratore.

[1] http://www.skype.com/
[2] https://docs.google.com/

Esistono poi altre forme di collaborazione tra cui vale la pena ricordare il controllo remoto, con il quale è possibile controllare a distanza un dispositivo attraverso un altro, il display sharing, che consiste nella possibilità per un utente di condividere con un altro utente ciò che viene visualizzato sullo schermo del proprio computer e la visualizzazione sincronizzata di una pagina Web, per fare in modo che gli utenti, accedendo alla pagina da browser diversi su dispositivi diversi, abbiano l'interfaccia utente relativa alla pagina in questione sincronizzata in tempo reale.
Il tipo di collaborazione via Web su cui invece pone particolare attenzione questa tesi è la realizzazione collaborativa di software attraverso la condivisione di un ambiente di sviluppo. La diffusione dei servizi cloud ha spinto anche il mondo degli IDE verso questa direzione. Recentemente si sta quindi assistendo allo spostamento degli IDE da ambienti desktop ad ambienti Web. Questo è determinante per quanto riguarda gli aspetti legati alla collaborazione perchè permette di sfruttare tutti i vantaggi del cloud per dotare questi sistemi di chat, integrazione con i social network, strumenti di editing condiviso e molte altre funzionalità collaborative. Questi IDE sono detti browser-based in quanto i servizi che mettono a disposizione sono accessibili via Web tramite un browser. Ne esistono di diversi tipi e con caratteristiche molto diverse tra di loro. Alcuni sono semplici piattaforme sulle quali è possibile effettuare test di codice o utilizzare tutorial forniti per imparare nuovi linguaggi di programmazione; altri invece sono ambienti di sviluppo completi dotati delle più comuni funzionalità presenti in un IDE desktop, oltre a quelle specifiche legate al Web.
Dallo studio di questi ambienti di sviluppo di nuova generazione è emerso che sono pochi quelli che dispongono di un sistema di collaborazione completo e che non tutti sfruttano le nuove tecnologie che il Web mette a disposizione. Per esempio, alcuni sono dotati di editor collaborativi, ma non offrono un servizio di chat ai collaboratori; altri mettono a disposzione una chat e il supporto per la scrittura simultanea di codice, ma non sono dotati di sistemi per la condivisione del display. Dopo l'analisi dei pregi e dei difetti della

collaborazione fornita dagli strumenti presi in considerazione ho deciso di realizzare delle funzionalità collaborative inserendomi nel contesto di un IDE browser-based chiamato InDe RT sviluppato dall'azienda Pro Gamma SpA.

Contributo

La struttura di InDe RT era adatta ad essere ad essere estesa per supportare collaborazione, ma queste funzionalità non erano implementate. Lo scopo di questa tesi è stato quindi quello di dotare InDe RT di un sistema di collaborazione in tempo reale. L'implementazione presenta due tipi diversi di applicazione: da un lato abbiamo la collaborazione a livello di IDE, che permette a due o più utenti di poter sviluppare uno stesso progetto software contemporaneamente, condividendo lo stesso spazio di lavoro, i dati e le informazioni all'interno di esso; dall'altro abbiamo un tipo di collaborazione a livello di applicazioni sviluppate utilizzando l'IDE che consiste nella possibilità per più utenti di utilizzare un'applicazione condividendo l'interfaccia utente e i dati in essa contenuti. Questo si traduce praticamente nell'esecuzione di un'istanza dell'applicazione sul browser di ogni utente che partecipa alla collaborazione. Il risultato di questa collaborazione, sia nel caso riguardi l'IDE sia nel caso di un'applicazione, sarà che tutti gli utenti che vi partecipano avranno l'interfaccia sincronizzata e vedranno le stesse cose nonostante utilizzino browser diversi in esecuzione su macchine diverse. Ho infine integrato nella collaborazione il supporto WebRTC (vedi paragrafo 3.3) e, relativamente alla collaborazione tra applicazioni, un sistema di mascheramento dei dati sensibili (paragrafo 3.4).

Per testare l'affidabilità e la scalabilità del sistema di collaborazione realizzato ho effettuato una serie di test volti ad analizzare il carico della rete durante le sessioni multiutente. In particolare ho focalizzato la mia attenzione al caso di sessioni di collaborazione con videochiamata multipla tramite WebRTC, cioè situazioni in cui durante una collaborazione tra molti utenti venga avviato un collegamento audio/video tra tutti i partecipanti. Ho notato come l'impatto del collegamento WebRTC audio/video sulla banda

a disposizione sia notevole e peggiori in maniera sostanziale le performance della collaborazione. Per questo motivo si è deciso di eliminare il video dalle videochiamate che coinvolgano più di due utenti, facendo in modo che in questo caso il collegamento avvenga solo tramite audio.

Struttura del documento

La tesi è strutturata nel modo seguente: nel secondo capitolo viene fornita una visione generale della collaborazione via Web, spiegando in cosa consiste e introducendo diverse forme di collaborazione real-time insieme agli strumenti che le rendono possibili.

Nel terzo capitolo viene illustrato in che modo la collaborazione real-time viene realizzata nel progetto alla base di questo lavoro di tesi e in come è stata sfruttata l'analisi dei sistemi collaborativi esistenti realizzata nel capitolo precedente ai fini dell'implementazione.

Il quarto capitolo è dedicato ai dettagli implementativi e alla spiegazione di tutti i moduli esterni utilizzati.

Il quinto capitolo contiene i dettagli relativi ai test eseguiti per la valutazione dei moduli realizzati.

Infine le conclusioni riassumono il lavoro svolto e rivolgono uno sguardo ai possibili sviluppi futuri.

Capitolo 2

Sistemi di collaborazione Web in tempo reale

Nel corso degli ultimi anni la popolarità del World Wide Web, o semplicemente Web, è cresciuta in maniera esponenziale. L'utilizzo di questo servizio è ormai da tempo entrato a far parte della quotidianità sia in ambito personale che lavorativo. La sua diffusione su scala globale, insieme all'avvento di dispositivi come gli smartphone che consentono di avere il WWW sempre a portata di mano, ha permesso di abbattere le barriere spaziali e temporali tra le persone. Questo ha aperto scenari molto interessanti come ad esempio la possibilità per gli utenti di condividere risorse e collaborare in tempo reale allo svolgimento di un'attività su di esse anche trovandosi in posti diversi gli uni dagli altri. Il termine risorsa, nel contesto di una collaborazione, può riferirsi a oggetti di varia natura: un documento, il display di un dispositivo, un canale di comunicazione o un software. Ecco quindi che la collaborazione real-time tra due o più utenti attraverso il Web si può presentare sotto diverse forme. Essa infatti può consistere:

- nell'editing simultaneo di un documento digitale;
- in un collegamento in videochat;
- nella condivisione dello schermo del proprio dispositivo;

- nella visualizzazione sincronizzata di una pagina Web;
- in un servizio di assistenza remota;
- nello sviluppo collaborativo di un software.

Nei paragrafi successivi verrà data una panoramica dei principali tool collaborativi esistenti per ognuna delle forme di collaborazione sopra citate.

2.1 Editor collaborativi

Nell'ambito della collaborazione sui documenti digitali uno degli strumenti più importanti e più utilizzati al giorno d'oggi è sicuramente Google Docs. Si tratta di un tool collaborativo per l'editing di documenti in tempo reale. I documenti possono essere condivisi ed editati da molti utenti contemporaneamente, i quali possono vedere le modifiche carattere per carattere mentre i collaboratori le effettuano. Non c'è un modo per evidenziare in tempo reale le modifiche fatte da un particolare utente durante una sessione di scrittura simultanea, ma comunque la posizione corrente dell'utente all'interno del documento condiviso è rappresentata da un cursore di un colore specifico che permette ad ogni collaboratore di avere una mappa della posizione degli altri sul documento. È presente poi una chat come supporto alla collaborazione e un log delle revisioni che permette agli utenti di vedere le modifiche apportate al documento da ciascun collaboratore.
Di recente anche Microsoft si è attrezzata per competere nel campo della condivisione in tempo reale realizzando Microsoft Office Online[1]. È la versione online del tradizionale e famoso pacchetto Office, comprendente software come Word, Excel e PowerPoint. Il suo utilizzo risulta pressochè identico per l'utente rispetto alla versione desktop, ma offre tutti i vantaggi legati al fatto di operare in rete: salvataggio dei documenti sul cloud, possibilità di condividere fogli di calcolo e presentazioni e di collaborare in tempo reale al loro editing.

[1] https://www.office.com/

2.2 Videochat

La videochat è la forma di collaborazione più diffusa. Viene effettuata attraverso sistemi che utilizzano la tecnologia VoIP (Voice over IP). Con VoIP si intende un insieme di protocolli che permettono di effettuare videochiamate sfruttando una connessione Internet. Tra i principali software che fanno uso di questa tecnologia ci sono applicazioni come Skype o Viber[2] che oltre a permettere il collegamento audio/video permettono anche la chat testuale. Il VoIP presenta però uno svantaggio. La videochat viene realizzata attraverso software o plug-in dedicati che fanno uso di questa tecnologia. Questo implica che due o più utenti che vogliano collegarsi in videochiamata debbano necessariamente installare il software o plugin in questione sul proprio dispositivo a volte con problemi di compatibilità tra versioni diverse o compatibilità tra la versione del software e il sistema operativo.
I recenti progressi nel campo delle tecnologie Web hanno permesso di sviluppare sistemi innovativi per la videochat. Nel 2011 Google ha rilasciato un progetto opensource per permettere la comunicazione diretta tra browser, noto come Web Real Time Communication[3] (WebRTC).

WebRTC
WebRTC è una tecnologia HTML5 che si propone di offrire agli sviluppatori Web degli strumenti integrati nel browser per inserire facilmente servizi di scambio di stream in tempo reale all'interno delle loro pagine Web. Per esempio servizi di comunicazione audio e video, di condivisione di video in tempo reale e condivisione dello schermo. WebRTC consente anche lo scambio di dati browser-to-browser come per esempio messaggi di chat e trasferimento di file. In termini pratici, per gli utilizzatori della rete, questo significa che non sarà più necessario scaricare, installare e configurare manualmente un'applicazione o usare un plug-in proprietario nel browser. Tutte le funzioni multimediali come i codec e la gestione degli stream scambiati sono integrate

[2]http://www.viber.com/
[3]http://www.Webrtc.org/

nativamente dagli sviluppatori dei browser e sono rese disponibili ai programmatori di applicazioni Web attraverso delle API [3]. Attualmente la tecnologia WebRTC è supportata dai seguenti browser (il numero rappresenta la versione a partire dalla quale è presente il supporto WebRTC):

- browser desktop:
  - Google Chrome 23;
  - Mozilla Firefox 22;
  - Opera 18.
- Android:
  - Google Chrome 28;
  - Mozilla Firefox 24;
  - Opera Mobile 12.
- iOS:
  - Bowser.

2.3 Strumenti di screen sharing

Un altro modo per effettuare la collaborazione consiste nella condivisione del display con un utente remoto. È uno dei metodi più utilizzati in quanto spesso si ha necessità che il collaboratore segua visivamente le operazioni che vengono effettuate dall'altro capo del collegamento, intervenendo quando necessario. In questo ambito uno dei software più importanti è TeamViewer[4]. TeamViewer realizza il controllo remoto, una tecnica che permette di mettere in comunicazione due dispositivi distanti tra loro consentendo ai collaboratori di lavorare sia su quello che hanno davanti che su quello lontano da essi. Quindi, nel caso di TeamViewer, lo screen sharing è un servizio derivato

[4]http://www.teamviewer.com/

da funzionalità più complesse come il controllo remoto. TeamViewer offre una collaborazione a tutto tondo permettendo, oltre al controllo remoto, lo scambio di file tra i dispositivi connessi e la possibilità di collegamenti in videochat.
Anche Skype fornisce un sistema di screen sharing come supporto alle videochiamate. Nel caso di Skype però la condivisione dello schermo è solo a livello di visualizzazione in quanto il collaboratore non ha possibilità di apportare modifiche a ciò che vede.

2.4 Strumenti per la visualizzazione collaborativa

Il concetto di collaborazione real-time trova applicazione anche in una vasta gamma di servizi offerti attraverso la rete che non riguarda la creazione o la modifica di risorse in senso stretto. È il caso, per esempio, delle applicazioni di e-learning e di e-shopping, o dei giochi collaborativi. Per questi servizi si parla di visualizzazione collaborativa. La collaborazione consiste infatti nella condivisione del browser tra i dispositivi degli utenti collegati e nella sincronizzazione dell'interfaccia utente. Lo scopo è quello di fornire agli utenti una visualizzazione condivisa del browser in modo da dare l'impressione che essi si trovino uno accanto all'altro e stiano osservando la pagina Web aperta sul browser attraverso un unico dispositivo. Alcuni framework che realizzano questo meccanismo sono PolyChrome [4] e Really Easy Displays (RED) [5].

PolyChrome
PolyChrome è uno strumento che consente di creare visualizzazioni collaborative Web-based su dispositivi multipli. Il framework supporta la visualizzazione condivisa delle applicazioni Web senza costi di migrazione (si parla quindi di browser distribuito). Fornisce inoltre un'API per sviluppare nuove applicazioni Web che possono sincronizzare lo stato della propria interfac-

cia utente su dispositivi multipli per supportare la collaborazione sincrona e asincrona. Poichè è scritto interamente in JavaScript, i dispositivi che partecipano alla collaborazione non hanno bisogno di nessun software particolare, a parte di un browser moderno. L'interazione e la sincronizzazione tra i dispositivi che utilizzano PolyChrome viene realizzata attraverso un collegamento peer-to-peer sicuro mentre le informazioni persistenti come dettagli di login, preferenze di configurazione del display e dettagli sulle condivisioni precedenti sono gestite da un server dedicato. PolyChrome offre inoltre la possibilità di salvare la storia di ogni collaborazione, costituita dalle interazioni tra gli utenti le quali sono rappresentate da operazioni. Combinato con lo stato iniziale dall'applicazione o del sito Web, il log delle interazioni è utile per sincronizzare i dispositivi all'interno dell'ambiente collaborativo e per permettere di riprodurre operazioni precedentemente effettuate.

Really Easy Displays
RED è una piattaforma Web-based il cui scopo è quello di facilitare le interazioni tra dispositivi e applicazioni. Fornisce una singola astrazione per contenuti e interazioni tra tipi di display diversi, flussi di dati e modalità di interazione e permette agli sviluppatori di creare applicazioni multi-display abilitando la condivisione del Document Object Model (DOM) tra i display. Il framework RED si basa su tre livelli:

- un'infrastruttura di supporto che fornisce un'adeguata ontologia, protocolli e gestione dei dati;

- un container per l'applicazione, Web-based o nativo di un display, che fornisce contestualizzazione del flusso di interazioni e accesso alle funzionalità native di un display;

- un livello di manipolazione del DOM basato su JavaScript che fornisce controllo dei display e un facile accesso ai dati.

Tramite questa struttura gli elementi di una pagina Web condivisa su più dispositivi, che possono consistere in testo, immagini, video ed eventi di interazione, risultano sincronizzati permettendo l'interoperabilità.

2.5 Strumenti per l'assistenza remota real-time

La collaborazione real-time ha permesso di dare un nuovo volto anche ai servizi di assistenza remota. Sono stati realizzati degli strumenti attraverso i quali un utente può avere un contatto diretto con l'assistente. Il caso più eclatante è rappresentato dal servizio MayDay[5] di Amazon. Questo servizio, accessibile esclusivamente dai tablet Kindle Fire HDX, consente di vedere la persona che effettua il supporto tecnico remoto in videochat attraverso una piccola finestra sullo schermo e visualizza anche lo schermo dell'utente sul dispositivo dell'assistente. In questo modo il tecnico ha la possibilità di guardare e vedere le operazioni che l'utente compie, toccare lo schermo e annotarlo attraverso l'interfaccia per indicare all'utente un'operazione da compiere. Il servizio di assistenza pensato da Amazon è disponibile 24 ore su 24 per 365 giorni all'anno. Per attivare la richiesta di assistenza basta premere il pulsante Mayday e un esperto Amazon in pochissimi secondi apparirà in video per ascoltare e risolvere i problemi dell'utente. Il collegamento video avviene però solo in una direzione, quella dell'utente. Egli infatti ha la possibilità di vedere e sentire l'assistente, ma quest'ultimo è collegato con l'utente solo via audio, nel rispetto della privacy di chi ha richiesto assistenza.
Un recentissimo e ancora in fase di sviluppo sistema di assistenza clienti real-time è il nuovo servizio Device Experts di Google. Consiste in un servizio di videochat live, rivolto a coloro che vogliono acquistare smartphone o tablet Google, che permette agli utenti di chiedere direttamente a degli assistenti informazioni sui prodotti prima di procedere all'acquisto. Si basa sull'infra-

[5]http://www.amazon.com/gp/help/customer/display.html?nodeId=201540070

struttura di Hangouts[6] e attualmente l'unico modo per accedervi è attraverso la sezione Dispositivi di Google Play. L'evoluzione di questo prototipo si pensa possa essere un servizio chiamato Google's virtual Genius Bar che prevede l'ingresso nei negozi al dettaglio attraverso help desk virtuali che accrescano l'esperienza di acquisto. Allo stato attuale non è ancora chiaro in che modo questo servizio verrà attuato, ma una soluzione potrebbe essere costituita da display con accesso diretto alla videochat con un assistente posizionati nella sezione dello store dove sono collocati i dispositivi Google.

2.6 Ambienti di sviluppo browser-based

La collaborazione real-time apre scenari interessanti anche per quanto riguarda il mondo dello sviluppo software, in particolare quello legato alle applicazioni Web. In quasi tutti i campi, la risposta ai problemi di comunicazione globale è stato il Web. Infatti, in un tempo relativamente breve, il Web è diventato la piattaforma per tutti i tipi di applicazioni e consente la collaborazione in tempo reale in forme e modi che sarebbero stati difficili da immaginare alcuni decenni fa. Recentemente, le funzionalità collaborative del Web sono state ulteriormente arricchite con una nuova invenzione, i social media. Facebook, Linkedin e altri servizi simili ci consentono di rimanere in contatto in tempo reale con amici, colleghi e persone di ogni parte del mondo che condividono le nostre passioni. Ma mentre gli utenti finali delle applicazioni si stanno massivamente spostando dal desktop al browser, la maggior parte degli sviluppatori lavora ancora con IDE desktop come Eclipse[7] o Visual Studio[8] [6]. Gli IDE desktop sono strumenti ad uso del singolo. Servono cioè ai singoli programmatori a comprendere e modificare il progetto software su cui il team di sviluppo di cui fanno parte sta lavorando, ma non permettono loro di sfruttare la conoscenza che altri membri del team possono avere riguardo la progettazione e la realizzazione del progetto. Lo

[6] https://plus.google.com/hangouts
[7] https://eclipse.org/
[8] http://www.visualstudio.com/

sviluppo di un software è invece un lavoro collettivo in cui il coordinamento e la comunicazione tra team di sviluppatori e tra gli sviluppatori all'interno di ogni singolo team è fondamentale per portare a termine il prodotto [7]. La limitazione degli IDE desktop risiede proprio nella loro incapacità di fornire coordinamento e collaborazione nello sviluppo. Il raggiungimento di questi obiettivi viene di solito perseguito avvalendosi di strumenti di document sharing e screen sharing simili a quelli citati sopra che rendono però la collaborazione molto complessa e spesso inefficiente.
In risposta a questi problemi negli ultimi anni sono nati degli IDE browser-based che trasformano l'IDE desktop in una serie di servizi integrati accessibili via browser. Questi IDE, come molti siti Web, sono solitamente composti da due parti: un frontend e un backend. Il frontend di solito è scritto in JavaScript e utilizza delle API Web per comunicare con il backend, anche se in alcuni casi un'applicazione desktop può agire da frontend e communicare con il backend senza la necessità di un browser. Il backend si occupa della creazione, del salvataggio e dell'apertura dei file, così come di eseguire comandi da terminale se l'IDE li supporta [8]. Questi IDE di nuova generazione, oltre a permettere allo sviluppatore di continuare ad utilizzare le funzionalità di un IDE tradizionale, offrono una moltitudine di vantaggi:

- l'unico strumento di cui il programmatore ha necessità per la sua attività di sviluppo è un browser. Quindi non si deve preoccupare di questioni come l'installazione, la configurazione e l'aggiornamento dell'ambiente di sviluppo all'ultima versione disponibile;

- permettono portabilità e continuità. Lo stato dell'IDE può essere salvato e riaperto su un'altra macchina, consentendo al programmatore di poter continuare il suo lavoro di sviluppo anche senza avere a disposizione il proprio dispositivo;

- lo spazio di lavoro è costituito da un ambiente singolo e centralizzato in cui una moltitudine di persone può sviluppare, editare ed eseguire

debug simultaneamente. Pertanto è possibile realizzare funzionalità che permettano la collaborazione in tempo reale;

- la possibilità di pubblicare applicazioni sul Web è facile da offrire come funzionalità dell'IDE.

Gli IDE Web presentano però anche alcuni svantaggi che possono essere riassunti nei seguenti punti:

- è indispensabile disporre di una buona connessione Internet;
- è necessario gestire i problemi legati alla sicurezza;
- ci sono possibilità di caduta temporanea del server.

Gli IDE browser-based rilasciati recentemente sono molti e di diverso tipo. Si va dai semplici tutorial per linguaggi di programmazione a soluzioni complete per lo sviluppo e il deploy di applicazioni Web. Possono essere divisi in due grandi categorie: gli IDE che si concentrano solo sull'aspetto client-side dello sviluppo Web e quelli che invece includono anche funzionalità server-side. Gli IDE del primo gruppo permettono la programmazione in JavaScript, HTML e CSS e di solito non costituiscono strumenti di sviluppo completi. Si tratta principalmente di framework più o meno complessi attraverso i quali effettuare test e debug sul codice. Uno di questi è jsFiddle[9] che permette di creare dei progetti chiamati fiddle per eseguire test su codice HTML, CSS e JavaScript allegando ad ogni fiddle anche alcuni dei più noti framework JavaScript. Offre anche la possibilità di salvare i fiddle creati e di condividerli con altri utenti.
Oltre ai linguaggi per il Web, alcuni di questi strumenti permettono la programmazione scegliendo fra un'ampia gamma di linguaggi e fornendo funzionalità di compilazione nel caso di linguaggi compilati. Un esempio è costituito da Coding Ground[10], un tool che permette di creare progetti utilizzando una

[9] http://jsfiddle.net/
[10] http://www.tutorialspoint.com/codingground.htm

vastissima selezione di linguaggi di programmazione e di compilarli ed eseguirli online. Offre anche la possibilità di condividere i progetti come servizio di collaborazione.
Sempre all'interno di questa categoria ci sono poi degli IDE che permettono la realizzazione di applet Java, piccole applicazioni scritte in Java che vengono eseguite attraverso la Java Virtual Machine direttamente su una pagina Web. È il caso di JavaWIDE[11] il quale include inoltre funzionalità come il supporto alle modifiche in concorrenza sul codice, una code base comune per tutti gli utenti, la gestione delle revisioni, codice sorgente annotato e con link multimediali, un' API Java integrata e molte altre funzionalità.
Gli IDE con il supporto server-side sono invece molto più complessi e potenti. Essi sfruttano tutte le potenzialità del cloud per realizzare ambienti integrati di sviluppo dotati anche di funzionalità di collaborazione. La maggior parte di questi strumenti si serve di Node.js[12] il quale rappresenta il modo più comune di includere funzionalità server-side in un'applicazione Web. Si tratta di un framework che permette allo sviluppatore di scrivere codice server-side in JavaScript.
Tra gli IDE che includono funzionalità server-side vale la pena citare e descrivere brevemente Cloud9[13], Koding[14] e Codio[15].

Cloud9

Cloud9 è un IDE Web molto evoluto che consente di sviluppare applicazioni attraverso l'uso di HTML, CSS, JavaScript, PHP, Java, Ruby e moltissimi altri linguaggi. Include molte funzionalità comunemente disponibili negli IDE desktop come per esempio l'highlighting della sintassi, l'abilità di eseguire e debuggare il codice, gli shortcut da tastiera, il drag and drop, il completamento automatico del codice e il rilevamento degli errori. Gli errori

[11] http://www.javawide.org/
[12] http://nodejs.org/
[13] https://c9.io/
[14] https://koding.com/
[15] https://codio.com/

di sintassi sono evidenziati immediatamente nel browser prima che il codice venga salvato sul server, velocizzando il ciclo di debug. Il fiore all'occhiello di Cloud9 è costituito da un potente terminale SSH che permette agli sviluppatori di eseguire facilmente comandi di shell e di vedere l'esecuzione del codice server-side scritto attraverso Node.js. Cloud9 supporta inoltre i sistemi di controllo di versione più conosciuti come Git, Mercurial e SVN. È uno degli IDE browser-based più popolari in quanto è considerato moderno e sicuro. Inoltre Cloud9 non solo permette agli sviluppatori di scrivere codice in maniera collaborativa, ma mette a disposizione una chat permettendo loro di collaborare in tempo reale.

Koding

Conosciuto precedentemente come Kodingen, Koding si propone di fornire un modo semplice per sviluppare e mettere in produzione applicazioni Web. Supporta quasi tutti i linguaggi di programmazione e librerie tra cui PHP, Python, Perl e JavaScript insieme a Django, Ruby e Node.js. Presenta un editor dotato di tutte le caratteristiche degli editor comuni, come numeri di linea, autoindentazione e autocompletamento. Supporta inoltre il multicursore che permette di effettuare modifiche multiple simultaneamente. Si tratta di un IDE molto avanzato che oltre a permettere lo sviluppo di software fornisce un ambiente di e-learning collaborativo dove gli studenti possono condividere il proprio lavoro e imparare in tempo reale. Una delle sue caratterstiche principali è la presenza di un terminale integrato che risulta essere uno dei più veloci e responsive tra tutti quelli presenti nei framework simili e supporta ogni programma terminal-oriented, database e compilatori. Koding integra inoltre funzionalità social media e ha una vasta comunità per il supporto ai progetti, che siano pubblici o privati.

Codio

Codio è uno degli IDE browser-based più recenti. È una piattaforma di sviluppo che presenta tutta la potenza di un IDE desktop unita alla semplicità e

all'utilizzabilità che gli sviluppatori si aspettano dalle moderne applicazioni browser-based. Permette agli sviluppatori di creare applicazioni front-end scritte in JavaScript, HTML5 e CSS e applicazioni back-end attraverso Node.js. Codio si differenzia dagli altri strumenti simili per il fatto di offrire un server virtuale dedicato per ogni progetto chiamato box. Le box sono istanze di Ubuntu su cui sono preinstallati Node.js, Ruby, Python, C, Java e servizi di controllo della versione come Git, Mercurial e SVN. Un'altra caratteristica che lo rende leggermente diverso dai suoi competitor è un meccanismo che permette di scrivere il codice HTML in maniera più compatta, lasciando alle funzionalità dell'editor il compito di trasformarlo in HTML standard. Codio supporta la collaborazione attraverso un sistema che permette l'editing real-time del codice da parte di molti sviluppatori simile a Google Docs. Per completare la collaborazione, nel prossimo futuro si propone di realizzare un servizio di chat e videochat direttamente intregrato nell'IDE.

2.7 Considerazioni finali

Tutti gli strumenti sopra illustrati mettono in luce quante applicazioni diverse può avere la collaborazione real-time e quanti benefici apporta ai contesti su cui è applicata. In un mondo dominato da smartphone, tablet e altre tecnologie portatili e dalla possibilità e necessità di avere un costante accesso a Internet, essa apre nuovi scenari nell'interazione tra gli utenti e nell'utilizzo del Web come risorsa condivisa. Come si è visto però la maggior parte di questi tool sono specializzati solo su una particolare forma di collaborazione. La carenza di strumenti collaborativi che integrino diverse forme di collaborazione al loro interno spinge gli utenti a realizzare delle soluzioni "fai da te", attraverso l'utilizzo simultaneo di più strumenti separati. Per esempio durante l'editing condiviso di un documento o nell'utilizzo di un IDE collaborativo, tra i collaboratori può nascere la necessità di avere un contatto più diretto, di parlare e scambiarsi opinioni. Essi allora saranno costretti a ricorrere ai servizi di chat, ai social network o a software che per-

mettono collegamenti audio e video come Skype. In ogni caso si troveranno a dover utilizzare diversi tool contemporaneamente con il risultato di rendere la collaborazione più macchinosa e meno pratica.
Nel capitolo successivo verrà illustrata una soluzione di collaborazione real-time che integra diverse funzionalità collaborative applicata al caso particolare dello sviluppo software attraverso un IDE browser-based.

Capitolo 3

Collaborazione real-time in un IDE browser-based

Il progetto realizzato durante il lavoro di tesi affronta il problema della collaborazione real-time nello sviluppo software. Il suo obiettivo è quello di fornire supporto per la realizzazione di sessioni di collaborazione sia all'interno di un IDE che all'interno delle applicazioni sviluppate con esso. In particolare ho preso in considerazione l'IDE Instant Developer Real-Time (InDe RT) prodotto dall'azienda ProGamma S.p.A.

3.1 InDe RT: un ambiente di sviluppo per applicazioni Web

InDe RT è un IDE per applicazioni mobile e Web browser-based, cioè viene eseguito direttamente sul browser come un normale sito Web. In figura 3.1 si può vedere un esempio di come si presenta la piattaforma. Il modello architetturale che caratterizza InDe RT e le applicazioni Web che esso permette di sviluppare è di tipo Model View Controller distribuito: c'è quindi una struttura client-server in cui il server (che usa Node.js) detiene il controllo su una sorta di DOM remoto e su tutti gli oggetti logici che compongono il sistema. Il client invece è rappresentato dal browser il quale,

attraverso una Websocket[1], riceve dal server le informazioni necessarie a dare una rappresentazione fisica agli oggetti logici, cioè a creare l'interfaccia dell'IDE o dell'applicazione così come sarà vista dagli utenti e notifica al server i cambiamenti e le modifiche apportate dall'utente. Generalizzando questo meccanismo al caso di più di un client, avremo un modello in cui diversi browser condivideranno lo stesso insieme di oggetti logici con il risultato che una modifica apportata da uno di questi alla propria interfaccia utente sarà propagata su tutti gli altri.
Benchè possa essere utilizzato su qualunque browser, sia in ambiente mobile che non, InDe RT è una piattaforma ideata per operare principalmente sul browser Google Chrome. Per questo motivo durante la realizzazione del progetto mi sono concentrato maggiormente su questo specifico browser.

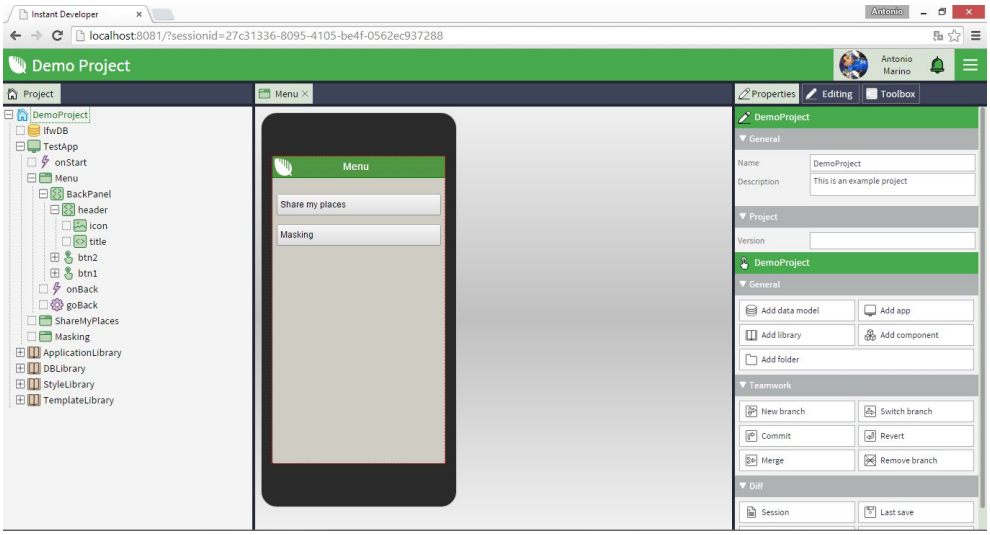

Figura 3.1: Interfaccia di InDe RT

Una visione d'insieme

La struttura di InDe RT era adatta a permettere la collaborazione, ma mancava un'implementazione che ne consentisse la gestione. Lo scopo di questa tesi è stato quindi quello di dotare InDe RT di un sistema di collaborazione

[1]http://socket.io/

in tempo reale. La figura 3.2 riassume la struttura di questo sistema. Nella parte superiore è rappresentato l'utilizzo dell'IDE per lo sviluppo di applicazioni Web. In questo contesto la collaborazione entra in gioco a due livelli rappresentati rispettivamente nei riquadri in basso a sinistra e a destra: da un lato abbiamo la collaborazione a livello di IDE, che permette a due o più utenti di poter sviluppare uno stesso progetto software contemporaneamente, condividendo lo stesso spazio di lavoro, i dati e le informazioni all'interno di esso; dall'altro abbiamo un tipo di collaborazione a livello di applicazioni sviluppate utilizzando l'IDE che consiste nella possibilità per più utenti di utilizzare un'applicazione condividendo l'interfaccia utente e i dati in essa contenuti. I rettangoli con bordo azzurro e verde rappresentano istanze dell'IDE e dell'applicazione. Il meccanismo di collaborazione, sia nel caso riguardi l'IDE, sia nel caso riguardi un'applicazione, consiste praticamente nella condivisione di una sessione tra più utenti che si traduce nell'esecuzione di un'istanza sul browser di ogni utente che partecipa alla collaborazione. All'interno di ogni sessione esiste un'istanza chiamata "owner" che è quella che ha avviato la condivisione e una o più istanze "guest", dipendenti dalla prima. Il risultato della condivisione della sessione sarà che tutti gli utenti che partecipano alla collaborazione avranno l'interfaccia sincronizzata e vedranno le stesse cose nonostante utilizzino browser diversi in esecuzione su macchine diverse. Questo concetto è sintetizzato in figura dalle frecce in entrambe le direzioni che collegano tra di loro tutte le istanze. Esse stanno appunto ad indicare lo scambio di dati e informazioni al fine di ottenere la sincronizzazione suddetta. A supporto della collaborazione ho inoltre integrato in entrambi i sistemi alcuni moduli, rappresentati dalle figure con bordo rosso. Una sessione condivisa prevede diversi attori che comunicano tra di loro. Per questo motivo ho sviluppato un modulo che si occupa della gestione dei ruoli dei collaboratori, ovvero si occupa di decidere in che modo essi possono operare all'interno della sessione. Come si può notare dalla figura, il gestore dei ruoli /'e reso disponibile solo all'istanza owner che cos/'i ha la possibilit/'a di controllare l'influenza che gli altri partecipanti alla ses-

sione possono avere nella collaborazione. Ho integrato poi un modulo per il supporto WebRTC grazie al quale è possibile effettuare videochiamate durante le sessioni di collaborazione. Infine, relativamente alla collaborazione a livello di applicazione, ho realizzato un sistema di mascheramento dei dati sensibili attraverso il gestore del data masking. Come nel caso del gestore dei ruoli, anche questo modulo è implementato solo sull'owner della sessione. In questo modo egli può decidere quali informazioni presenti nell'interfaccia dell'applicazione rendere disponibili ai collaboratori.
Nei paragrafi seguenti verrà illustrata più in dettaglio la collaborazione real-time sia all'interno dell'IDE che per quanto riguarda le applicazioni.

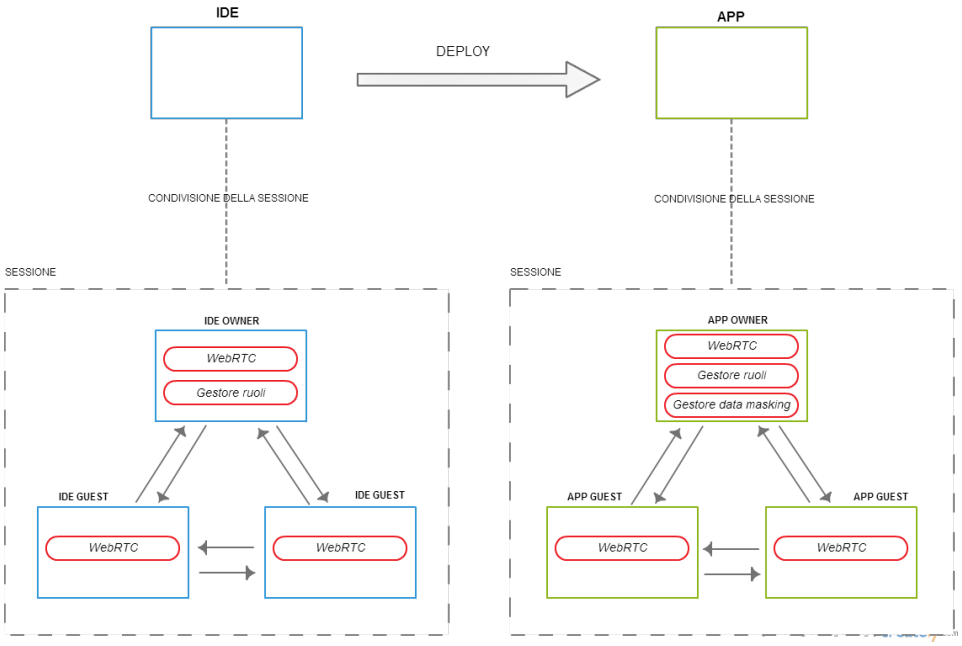

Figura 3.2: Sessioni di collaborazione

3.2 Supporto per la condivisione della sessione e per la gestione dei ruoli

Come detto, InDe RT è una piattaforma browser-based e per questo motivo il passo fondamentale per realizzare la collaborazione è la condivisione

della sessione fra i browser degli utenti che vogliono collaborare. Per sessione si intende una specifica esecuzione su un browser dell'IDE o di un'app sviluppata tramite questo, caratterizzata da un token che permette al server di identificarla univocamente. Quando ad una sessione partecipano più utenti si parla di sessioni multiutente. Le sessioni multiutente sono necessarie sia per permettere lo sviluppo software collaborativo usando l'IDE e sia per poter interagire in modo simultaneo nelle applicazioni.
Quando una risorsa o un insieme di risorse è condivisa, come avviene nelle sessioni, è importante definire quali sono le azioni che ogni utente può compiere in modo da poter distinguere quelli con più privilegi da quelli più limitati. Per questo motivo, all'interno delle sessioni, ad ogni utente viene assegnato un ruolo che definisce i diritti di cui gode.
Vediamo ora nel dettaglio come vengono condivise le sessioni, quali sono i ruoli e come vengono assegnati.

3.2.1 Condivisione della sessione dell'IDE

Il contesto in cui ci troviamo quando parliamo di sessione dell'IDE è quello di un utente che sta sviluppando un progetto utilizzando InDe RT. Infatti il browser su cui l'IDE è in esecuzione opera all'interno di una sessione che il server è in grado di identificare grazie ad un token.
La condivisione della sessione entra in gioco nel momento in cui nell'utente nasce la necessità o la volontà di condividere il proprio lavoro con un altro utente e quindi di permettere a quest'ultimo di collaborare con lui allo sviluppo del progetto. Per avviare la collaborazione l'utente deve invitare il collaboratore alla sessione, attraverso un comando dell'IDE. L'invito è rappresentato da un URL formato dal token identificativo della sessione e da un ulteriore token univoco per l'utente invitato. Questo token viene generato dal sistema e può essere comunicato a chi si vuole invitare utilizzando un qualche canale di comunicazione esterno, tipicamente l'email. In figura 3.3 si può vedere la generazione di un URL di connessione.

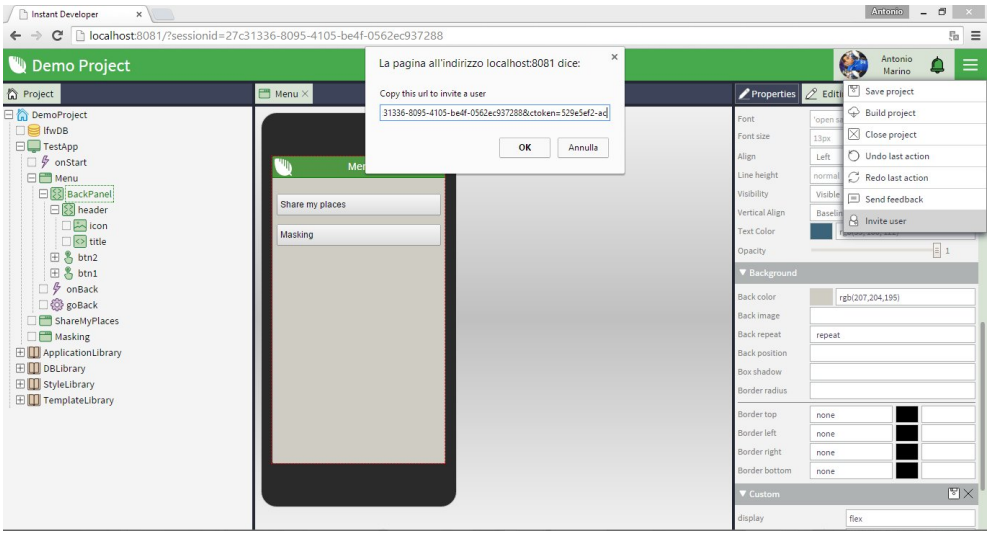

Figura 3.3: Generazione URL per invitare un utente

L'URL di connessione è univoco e questo implica che se si vuole invitare alla sessione più di un utente bisogna generare tanti URL quanti sono gli utenti collaboratori. Inoltre, per ragioni di sicurezza, non è riutilizzabile. In questo modo si evita di ammettere alla sessione utenti indesiderati che siano entrati in possesso dell'URL.

Quando l'utente remoto si collegherà a quest'URL il sistema, attraverso un componente server-side che analizza il token di sessione e il token che rappresenta l'utente invitato, permetterà al browser remoto di entrare a far parte della sessione e quindi di condividere il progetto. A questo punto ha inizio la collaborazione real-time. Tutti i browser che partecipano alla sessione avranno a disposizione l'IDE e il progetto sul quale si sta lavorando e le operazioni eseguite da uno di questi saranno propagate dal server su tutti gli altri ottenendo una sincronizzazione in tempo reale dello stato dell'IDE e del progetto.

La condivisione della sessione è la base su cui si fonda la collaborazione real-time. Da sola però non è sufficiente per sfruttarne al massimo le potenzialità. Per questo motivo ho realizzato una serie di funzionalità che permettono di rendere più efficace questo meccanismo. Innanzitutto è indispensabile che gli utenti possano comunicare fra di loro. A tale scopo ho realizzato una chat a

cui tutti i browser che partecipano alla sessione sono collegati (fig. 3.4).

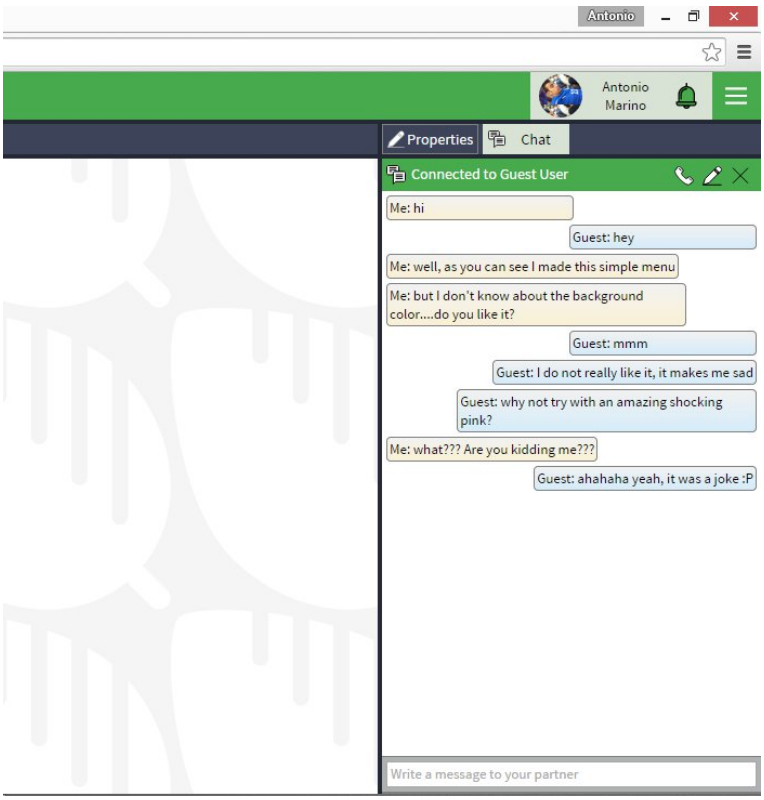

Figura 3.4: Chat di sessione

Ho realizzato inoltre un sistema di hint visuali per permettere ad un utente di avere una maggiore percezione di una modifica effettuata da un browser remoto. Questo vuol dire che quando un utente effettua operazioni quali le interazioni con l'albero degli elementi che compongono il progetto (espansione, contrazione, aggiunta ed eliminazione di un nodo) oppure modifiche alle proprietà di un componente del progetto, tutti gli utenti della sessione vedranno nel loro browser un effetto in corrispondenza del componente che ha subito la modifica che serve a rendere più evidente il suo cambiamento di stato. Un esempio di hint visuale è mostrato in figura 3.5.

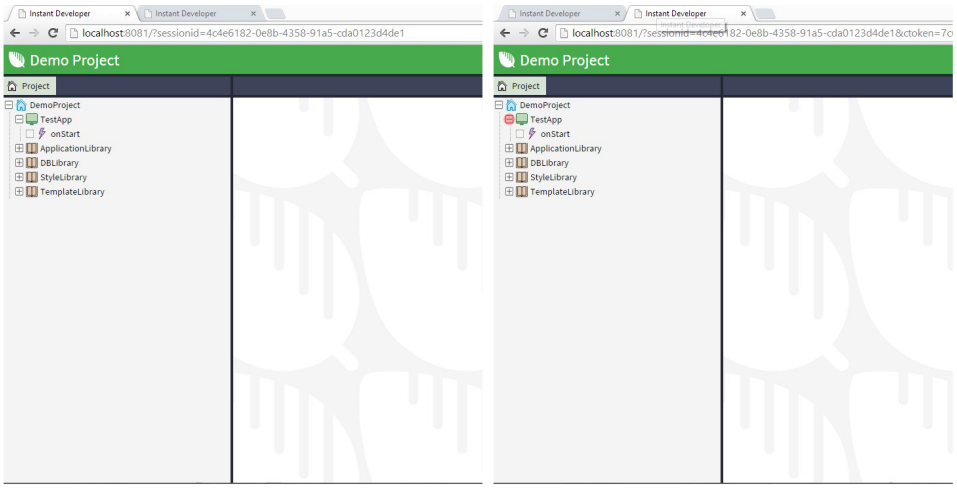

Figura 3.5: Effetto generato dall'espansione di un nodo dell'albero da parte dell'utente remoto

3.2.2 Gestione dei ruoli

Uno degli aspetti più importanti nella condivisione di una sessione è la gestione dei ruoli degli utenti che ne fanno parte. Assegnare un ruolo significa specificare privilegi e limitazioni riguardo le operazioni che un utente può effettuare. Poiché l'utente che ha avviato la sessione è il proprietario del progetto, ad egli deve essere dato il potere di controllo della sessione e degli utenti connessi, i quali di conseguenza risulteranno subordinati.
Gli utenti sono divisi in:

- owner: è colui che ha creato la sessione e quindi ne è il possessore;
- guest: tutti quelli che sono stati invitati dall'owner.

L'owner ha tutti i diritti, sia in lettura che in scrittura ed ha inoltre potere sulla vita dei guest: decide il loro ruolo (fig. 3.6) e può decidere quando disconnetterli dalla sessione. I guest invece sono dipendenti dall'owner, il quale appunto assegna loro il ruolo che specifica i loro diritti all'interno della sessione. Il ruolo di un guest può essere:

- read only: è il ruolo di default. Non dà permesso in scrittura, perciò qualunque modifica apportata al progetto non viene inviata al server e quindi viene persa;

- read write: stessi diritti dell'owner anche se è sempre dipendente da quest'ultimo.

Entrambi i ruoli che un guest può assumere comportano inoltre delle limitazioni nell'insieme dei comandi disponibili nel menu principale dell'IDE. Per un guest read only non è disponibile nessun comando, mentre nel caso di guest read write l'insieme dei comandi è parziale rispetto a quello visibile dall'owner della sessione, che invece dispone dell'insieme completo. Per esempio il comando per salvare il progetto è disponibile solo nel menu dell'owner. In figura 3.7 si possono vedere i tre menu diversificati.
Sempre nell'ottica di dare a chi ha creato la sessione il pieno controllo della collaborazione, anche il termine di quest'ultima è deciso dall'owner. Ho deciso di associare il termine di una sessione di collaborazione alla chiusura della chat in quanto questa è un elemento essenziale di una collaborazione. La chiusura della chat da parte dell'owner comporta la disconnessione di tutti gli utenti collegati e quindi la loro esclusione dalla sessione. Da parte sua, un guest ha la possibilità di abbandonare la sessione singolarmente sempre attraverso la chiusura della chat e senza influenzare in alcun modo il prosieguo della collaborazione.
È importante sottolineare come le politiche applicate dall'owner all'interno di una sessione di collaborazione non si riflettono sui guest singolarmente, ma li riguardano tutti. Per esempio, quando l'owner decide di dare il permesso in scrittura, questo viene dato a tutti i guest. O ancora, quando l'owner chiude la chat disconnette tutti i guest collegati. Si ha quindi una relazione di controllo uno a molti dall'owner verso i guest.

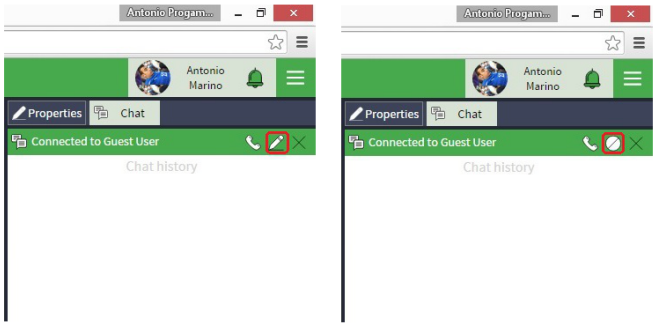

Figura 3.6: Impostazione del ruolo read only e read write

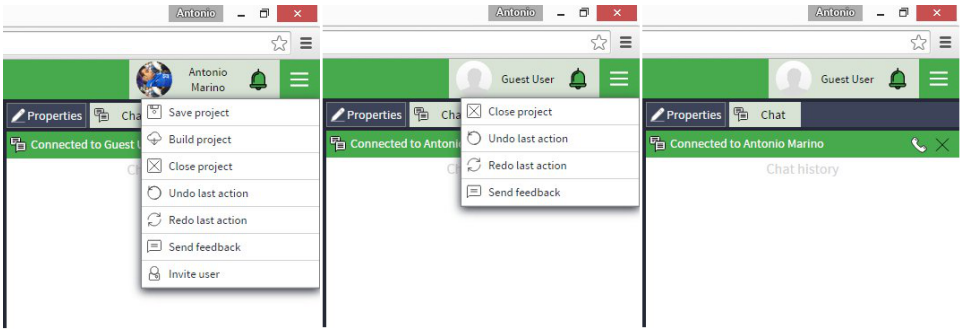

Figura 3.7: Da sinistra verso destra: menu owner, menu readwrite, menu readonly

3.2.3 Condivisione della sessione di un'applicazione

Un'altra modalità di collaborazione consiste nelle sessioni multiutente a livello di applicazione. Una sessione multiutente a livello di applicazione consiste nel condividere la sua esecuzione su tutti i dispositivi collegati alla sessione. Questo tipo di collaborazione può avere diversi casi d'uso interessanti, primo fra tutti l'assistenza remota. Un utente che ha bisogno di assistenza remota riguardo l'utilizzo di un'applicazione può chiedere all'assistente di collegarsi alla sua sessione e poiché l'esecuzione è condivisa, entrambi vedranno le stesse cose rendendo l'assistenza molto efficace. Un altro esempio di possibile caso d'uso è la creazione condivisa di un itinerario di vacanza attraverso un'applicazione che utilizzi una mappa.

La creazione e la gestione di una sessione di collaborazione a livello di applicazione avviene in maniera molto simile alla collaborazione a livello IDE. L'esecuzione dell'applicazione è associata ad una sessione che viene identificata dal server attraverso un id univoco. Anche in questo contesto per iniziare una collaborazione è necessaria la generazione di un URL da condividere con l'utente con cui si vuole collaborare. Questo URL viene generato dall'applicazione e ha le stesse caratteristiche di univocità e non riutilizzabilità dell'URL che permette ad un utente di collegarsi ad una sessione dell'IDE. è formato quindi da un token che rappresenta l'id della sessione e da un altro token che rappresenta il client che si collegherà. Queste due informazioni saranno utilizzate da un componente server-side per ammettere il client alla sessione. A differenza delle sessioni multiutente a livello IDE però, per le applicazioni la possibilità di generare un URL di connessione alla sessione non è presente come funzionalità di sistema, ma è fornita come metodo di libreria in modo da permettere allo sviluppatore di decidere se l'applicazione che sta sviluppando supporti o meno le sessioni multiutente.
Quando l'utente invitato alla sessione si connette inizia la collaborazione, che comporta l'esecuzione dell'applicazione sul dispositivo di ogni utente collegato. Con la presenza di più utenti si ha necessità di una politica di assegnazione e gestione dei ruoli degli utenti connessi per definire quanto potere ha ognuno di essi all'interno della sessione condivisa. Avremo quindi una suddivisione degli utenti in:

- owner: è colui che ha creato la sessione e gode di tutti i diritti, sia in lettura che in scrittura;
- guest: è un utente invitato dall'owner alla sua sessione. Può assumere due ruoli differenti: read only oppure read write.

Un guest read only non ha la possibilità di interagire con l'interfaccia dell'app, nel senso che le operazioni che effettua non vengono notificate al server e quindi non hanno effetto sull'app in esecuzione sui dispositivi degli altri utenti della sessione. Un guest read write invece ha gli stessi diritti dell'owner della

sessione. Il compito di assegnare i ruoli ai guest è delegato all'owner sempre nell'ottica di concedere il controllo totale sulla sessione a chi ne è il creatore.

3.3 Supporto WebRTC

Per migliorare la comunicazione e l'operatività tra gli utenti di una sessione collaborativa ho aggiunto la possibilità di effettuare videochiamate durante la collaborazione attraverso l'integrazione di WebRTC. In questo modo i collaboratori, oltre a condividere il proprio lavoro hanno la possibilità di vedersi e di sentirsi reciprocamente e quindi di avere un contatto diretto.

Ho realizzato il supporto WebRTC per la collaborazione in due forme differenti:

- videochiamate all'interno di una sessione IDE;
- videochat in una sessione di un'applicazione.

Per il momento si è deciso di inserire una limitazione nella videochat: nel caso di una collaborazione che coinvolga più di due utenti la videochat sarà disponibile esclusivamente in modalità audio, senza quindi la condivisione degli stream video . Questa scelta è stata fatta in quanto per rendere possibile la videochat multipla includendo anche il video ci sarebbe bisogno di un server esterno che effettui il multiplexing degli stream video, i quali altrimenti saturerebbero la banda diminuendo le prestazioni generali e rendendo inefficiente la collaborazione.
Esiste poi una terza forma di supporto WebRTC totalmente separata dal concetto di sessione di collaborazione. Ho realizzato infatti un componente che può essere integrato nell'applicazione in fase di sviluppo per permettere la videochat con altre applicazioni esterne che utilizzino la tecnologia WebRTC.
Vediamo di seguito i dettagli relativi al supporto WebRTC nelle tre forme appena descritte.

3.3.1 WebRTC all'interno di una sessione dell'IDE

All'interno dell'IDE è stata sfruttata la possibilità fornita da WebRTC di scambiare stream audio e video per consentire a due utenti di collegarsi in videochat. Durante una sessione condivisa l'instaurazione del collegamento fra due utenti avviene attraverso i seguenti passi:

- avvio della videochiamata cliccando il pulsante di chiamata;
- autorizzazione all'accesso alla Webcam e al microfono del dispositivo da parte del browser;
- scambio automatico delle credenziali tra i client;
- creazione del collegamento.

All'inizio della videochiamata gli stream audio e video dei partecipanti sono entrambi attivi. In qualunque momento ognuno dei partecipanti alla videochiamata ha la possibilità di spegnere o accendere la propria Webcam, attivare o disattivare il microfono e chiudere la videochiamata. è possibile eseguire queste operazioni tramite i controlli posti sotto la finestra del video remoto, cioè quello che viene inviato dall'altro partecipante. Ho ritenuto importante fare in modo che ogni utente sappia sempre se sta inviando il proprio stream video o meno. Per fare questo ho utilizzato l'immagine del profilo. Quando lo stream video è attivo l'immagine di profilo mostrerà il video catturato dalla propria Webcam.

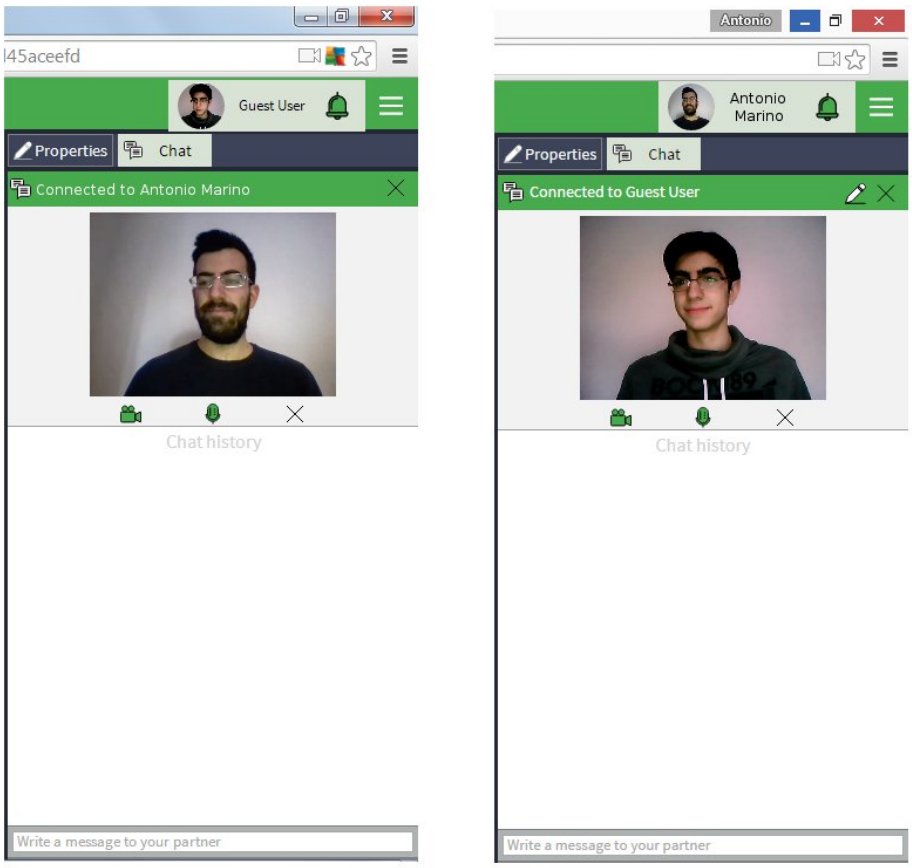

Figura 3.8: Videochiamata WebRTC in una sessione IDE

3.3.2 WebRTC all'interno della sessione di un'applicazione

All'interno di una sessione multiutente di un'applicazione la videochiamata è parte integrante della sessione stessa. Questo vuol dire che nel momento stesso in cui l'utente invitato si collega, il browser richiederà il consenso per accedere alla Webcam e al microfono e successivamente verrà avviata la videochiamata. Di default un client che si collega alla sessione fornisce sia gli stream audio che video, mentre l'owner fornisce solo lo stream audio. Questa scelta è stata fatta per venire incontro a situazioni reali. Un esempio tipico è rappresentato dall'assistenza remota. L'utente che richiede l'assi-

stenza remota invia un link all'assistente per collegarlo alla sua applicazione; quando partirà la videochiamata, l'utente vedrà e sentirà l'assistente, mentre quest'ultimo riceverà solo lo stream audio dell'utente. Sarà poi l'utente a decidere successivamente se inviare anche lo stream video. La finestra della videochat è dotata di alcuni controlli per abilitare e disabilitare il video, accendere e spegnere il microfono e chiudere la videochiamata. La chiusura della videochiamata da parte dell'owner comporta la disconnessione del client collegato, il quale ha anche la possibilità di uscire dalla sessione autonomamente chiudendo la videochiamata dalla sua parte.
In figura 3.9 è mostrato un esempio di videochiamata all'interno di una sessione di un'applicazione con gli stream video attivi per entrambi gli utenti.

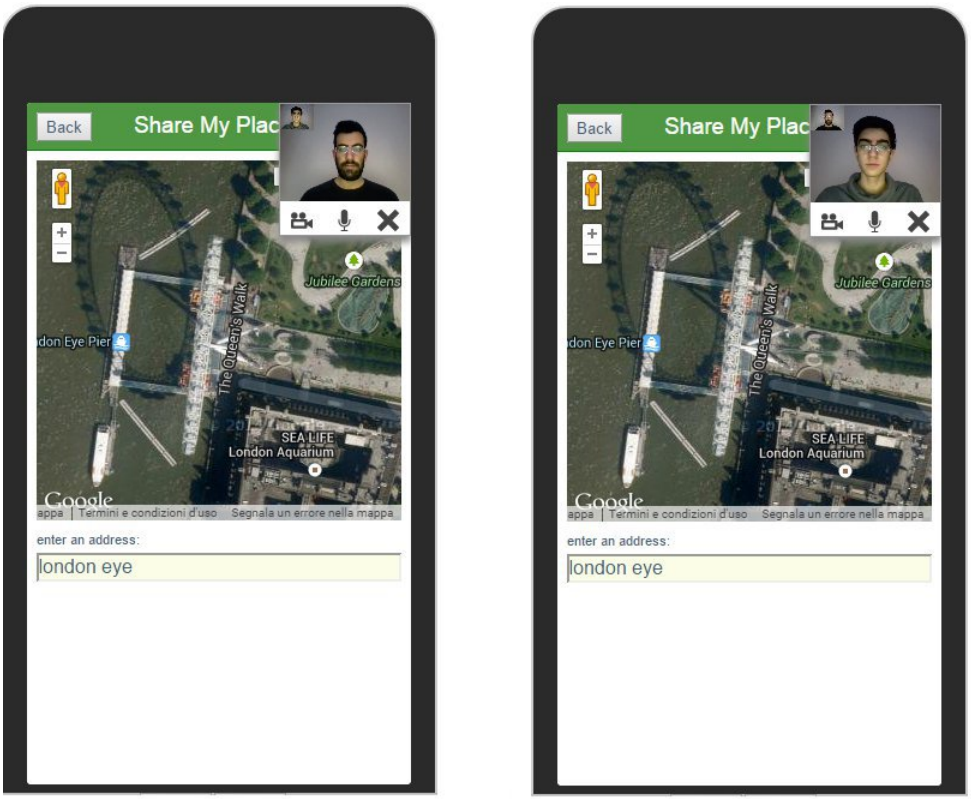

Figura 3.9: Videochiamata in una sessione di un'applicazione

3.3.3 Componente WebRTC per lo sviluppo delle applicazioni

L'utilizzo della tecnologia WebRTC è in costante aumento. Questo vuol dire che ci saranno sempre più applicazioni che la utilizzeranno. Ho pensato quindi che potrebbe essere utile per un'applicazione poter scambiare stream con qualunque altra applicazione che utilizzi lo stesso framework, senza che queste siano state realizzate attraverso lo stesso IDE o che venga creata una sessione condivisa. L'implementazione di un sistema che permetta ad un'applicazione di effettuare videochiamate non risulta di veloce realizzazione perché occorre prima realizzare l'infrastruttura e dotarsi di un server per permettere ai client di collegarsi.
Per rendere più agevole l'integrazione di WebRTC nelle applicazioni ho realizzato un componente che permette agli sviluppatori di gestire tutti gli aspetti legati alla gestione di una videochiamata in modo semplice. Il componente fornisce infatti supporto per avviare e chiudere una videochiamata, gestire le chiamate in arrivo e per definire le impostazioni della videochiamata. Inoltre viene fornito il server che permette di realizzare l'infrastruttura necessaria per il collegamento.

3.4 Data masking

Il data masking è un aspetto che riguarda le sessioni multiutente a livello di applicazione. All'interno di una sessione condivisa potrebbero esserci delle informazioni sensibili che l'owner non vuole condividere con gli utenti collegati. è il caso per esempio di un campo di testo, ma anche di un'immagine che non si vuole rendere disponibile dall'altro capo della collaborazione. Per permettere di mascherare i dati sensibili ho aggiunto delle funzionalità che consentono di nascondere alcune informazioni agli utenti che si collegano alla sessione.
Ogni oggetto dell'applicazione creato nell'IDE in fase di sviluppo può essere contrassegnato dal programmatore come dato sensibile. In questo modo

all'interno della sessione i dati contenuti in questi oggetti verranno mascherati. Per quanto riguarda i campi di testo o comunque tutti gli altri oggetti in cui il dato sensibile è rappresentato da una stringa (come per esempio il nome visibile di un pulsante), il mascheramento avviene trasformando la stringa originale in una stringa di asterischi. Alle immagini invece, viene applicato un effetto "blur". Questo effetto sfuoca l'immagine abbassandone la definizione e rendendola non identificabile. Ovviamente nel caso delle immagini il mascheramento è fittizio, in quanto viene realizzato semplicemente impostando una regola di stile. Ho scelto di realizzarlo in questo modo solo a scopo dimostrativo. In una fase successiva verrà reso effettivo impostando un'immagine di default in sostituzione dell'immagine originale quando quest'ultima rappresenta un dato sensibile.

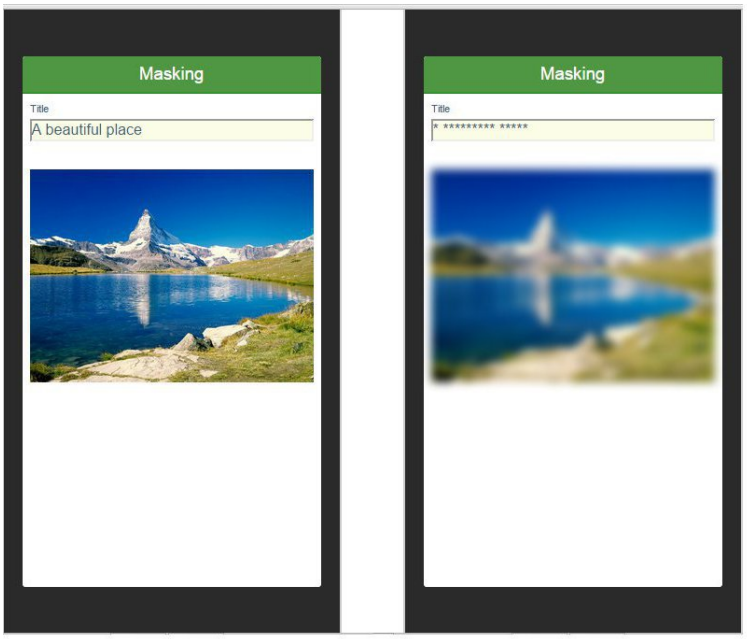

Figura 3.10: Effetto del mascheramento dei dati su una stringa e su un'immagine dalla parte dell'utente guest

Capitolo 4

Implementazione dei moduli

In questo capitolo verranno trattati in dettaglio l'implementazione delle sessioni condivise, l'integrazione di WebRTC nei diversi contesti in cui viene utilizzato e il mascheramento dei dati. Tutta l'implementazione è stata realizzata utilizzando il linguaggio di programmazione Web JavaScript.
Per quanto riguarda la parte legata a WebRTC, prima di approfondire il lavoro effettuato in questa tesi, è utile dare una panoramica dell'architettura e del funzionamento a basso livello di questa tecnologia, oltre a descrivere i moduli esterni utilizzati a supporto dell'implementazione.

4.1 L'infrastruttura di base

Come detto in precedenza, WebRTC permette la comunicazione diretta browser-to-browser per lo scambio di dati e di stream di diverso formato come file, stream audio e stream video. In gergo, gli endpoint di una comunicazione WebRTC vengono chiamati peer, quindi si parla anche di comunicazione peer-to-peer (P2P).
Per poter instaurare una comunicazione P2P è necessario che i peer coinvolti siano in grado di identificarsi l'uno con l'altro. Ogni peer sulla rete è dotato di due indirizzi IP, uno privato e uno pubblico. L'IP privato identifica il peer solo localmente, cioè solo all'interno del dominio in cui si trova, per esem-

pio all'interno di una rete aziendale. Per la connessione a Internet è invece necessario l'IP pubblico che permette al peer di essere identificato in modo univoco globalmente e quindi di essere visibile al di fuori della rete privata in cui si trova. Un peer che si connette a Internet però non conosce il proprio IP pubblico perchè l'operazione di trasformare l'IP privato in IP pubblico (e viceversa) viene eseguita dal router attraverso una tecnica chiamata Network Address Translation (NAT). Per permettere ai peer di conoscere il proprio indirizzo IP pubblico e quindi di poter instaurare una connessione P2P, WebRTC utilizza il framework Interactive Connectivity Establishment (ICE) [9].

ICE si avvale di un server che implementa il protocollo Session Traversal Utilities for NAT (STUN). STUN è un protocollo che funge da tool per altri protocolli nell'affrontare la questione del NAT traversal. Può essere utilizzato da un endpoint per determinare l'indirizzo IP e la porta assegnategli da un NAT. Il protocollo STUN funziona con molti NAT esistenti e non richiede loro comportamenti particolari [10].

Il server STUN viene utilizzato solo durante la fase di instaurazione del collegamento e una volta che la connessione è stata stabilita i dati fluiscono tra i peer in modo diretto. Esistono molti server disponibili che implementano il protocollo STUN. Google per esempio ne mette a disposizione alcuni che possono essere liberamente utilizzati nell'implementazione di servizi di connessione P2P.

Ci sono alcune situazioni in cui però una connessione diretta P2P non può essere stabilita. Questo succede quando uno o entrambi i peer si trovano sotto un tipo di NAT con cui il server STUN non funziona oppure quando il firewall blocca le richieste di connessioni dirette che provengono dall'esterno. In questi casi il framework ICE utilizza un server che implementa il protocollo Traversal Using Relays around NAT (TURN) [11]. Il server TURN è un nodo intermediario che agisce da relay nella comunicazione tra i peer. È stato pensato per essere usato da ICE nell'interfacciarsi con il NAT traversal, ma può essere utilizzato anche senza ICE.

Il protocollo TURN è un'estensione del protocollo STUN. Pertanto un server TURN li implementa entrambi. A differenza di un server STUN, il server TURN è necessario anche dopo la fase di instaurazione del collegamento. Esso infatti agisce da intermediario nello scambio di dati tra i peer che altrimenti non potrebbero comunicare. Poichè al momento non ci sono server liberamente utilizzabili già predisposti per essere usati come server TURN, per sviluppare un servizio di comunicazione P2P è necessario disporre di un server proprio e configurarlo in modo che implementi questo protocollo. Google mette a disposizione un'implementazione open source di un TURN server che consiste in un progetto chiamato rfc5766-turn-server[1]. Questo progetto è stato utilizzato per l'implementazione del supporto WebRTC realizzato in questo lavoro di tesi.

4.1.1 Il modulo PeerJS

A supporto dell'implementazione del sistema di videochiamate ho utilizzato lla libreria JavaScript PeerJS[2]. PeerJS utilizza l'implementazione WebRTC del browser per fornire delle API complete, configurabili e semplici da usare che permettono di realizzare la connessione peer-to-peer. Attraverso nient'altro che un ID, un peer può creare una connessione P2P con un peer remoto. PeerJS comprende al suo interno un modulo chiamato PeerServer che può essere installato sul proprio server e viene utilizzato per fornire ai peer l'id che essi useranno per instaurare la connessione P2P.
Questa libreria semplifica l'utilizzo delle API native di WebRTC attraverso due classi fondamentali: la classe Peer e la classe MediaConnection.
La classe Peer utilizza internamente il framework ICE per realizzare tutto il meccanismo di instaurazione della comunicazione di cui sopra, permettendo allo sviluppatore di dover fornire solo i riferimenti ai server TURN/STUN

[1] https://code.google.com/p/rfc5766-turn-server/
[2] http://peerjs.com/

in fase di creazione di un oggetto Peer. I metodi principali di questa classe sono:

- call(remoteId, localStream): permette di effettuare una chiamata verso il peer identificato dal remoteId, inviando a questo gli stream locali del chiamante. Restituisce un oggetto MediaConnection che rappresenta la chiamata;
- on("open", function(id)): è il listener sull'evento di connessione del Peer al server PeerServer. La sua callback fornisce come parametro l'id del Peer;
- on("call", function(call)): è il listener sulle chiamate in entrata. La callback fornisce un oggetto MediaConnection che rappresenta la chiamata in arrivo.

La classe MediaConnection permette invece di gestire lo scambio di stream multimediali tra i peer. I suoi metodi sono:

- answer(localStream): permette di rispondere ad una chiamata in entrata inviando i propri stream all'altro capo della comunicazione;
- close(): è il metodo che permette di terminare una chiamata;
- on("stream", function(remoteStream)): è il listener sulla ricezione degli stream inviati dal peer remoto, forniti come parametro dalla callback;
- on("close", function()): è il listener sulla chiusura della chiamata.

La corretta interazione tra i metodi di queste due classi permette di realizzare una videochiamata.

4.2 Sessioni collaborative a livello IDE

Analizziamo ora l'implementazione della collaborazione all'interno dell'IDE.

La fase iniziale di una collaborazione consiste nella condivisione della sessione. Quando l'owner utilizza il comando Invite user del menu principale dell'IDE vengono eseguite le seguenti operazioni:

- viene generato un token univoco;
- il token viene inviato al server sulla Websocket che rappresenta il canale di comunicazione tra server e client;
- il server riceve il token e lo salva in un array associato alla sessione dal quale lo ha ricevuto;
- viene creato e mostrato a video un URL con una query string formata da un parametro sessionid che rappresenta appunto l'id della sessione e un parametro ctoken che rappresenta il token associato al nuovo client.

L'URL rappresenta la chiave di accesso alla sessione. Infatti, quando un utente si collegherà all'URL, il server controllerà la presenza del ctoken nell'array associato al sessionid. In caso affermativo ammetterà l'utente attraverso il redirect all'indirizzo della sessione.
Il nucleo dell'implementazione della collaborazione è la classe JavaScript ChatView che implementa la chat di sessione. Quando la sessione diventa multiutente, il server invia il comando di creazione dell'oggetto ChatView a tutti i client. La chat opera in questo modo: un messaggio di chat inviato da un client sulla Websocket viene ricevuto dal server e da questo reinviato su tutti i client eccetto il mittente originale. Ho sfruttato allora la Websocket per la gestione della collaborazione tra gli utenti. Ho trasformato i messaggi di chat da semplici stringhe ad oggetti costituiti da un parametro type e un parametro content. In questo modo, alla ricezione di un messaggio il client esegue un metodo diverso in base al type del messaggio. I tipi di messaggio previsti sono i seguenti:

- chatMsg
- changeUserRole

- view
- closeChat
- remoteIdReq
- calleeReady
- closeCall
- disconnectClient

Vediamo ora in in che modo vengono sfruttati questi tipi nella realizzazione delle funzionalità legate alla collaborazione.

4.2.1 Gestione dei ruoli

La gestione dei ruoli consiste nel fare in modo che gli utenti guest di una sessione abbiano meno privilegi rispetto all'owner. Il potere di modificare il ruolo dei guest è dato all'owner attraverso un pulsante presente nell'interfaccia dell'IDE posizionato sopra la finestra di chat. Al click su questo pulsante viene inviato sulla Websocket un messaggio con il parametro type settato a "changeUserRole" e il parametro content settato a "ro", per assegnare agli utenti guest il ruolo readonly, o a "rw", per assegnare il ruolo readwrite.
Quando i guest ricevono il messaggio eseguono il metodo changeUserRole passandogli come parametro il content del messaggio, cioè il ruolo deciso dall'owner. Questo metodo si occupa di:

- settare la proprietà userRole del guest al valore presente nel campo content del messaggio;
- modificare il menu principale dell'IDE coerentemente con il nuovo ruolo. Il ruolo "rw" prevede la rimozione dei comandi Save project, Build project e Invite user mentre il ruolo "ro" prevede la rimozione di tutti i comandi dal menu;

- impostare a true la proprietà readOnly dell'oggetto che rappresenta il progetto. Questo blocca il sistema di eventi attraverso il quale il client notifica al server i cambiamenti avvenuti nella sua interfaccia grazie ad un controllo su questa proprietà effettuato lato server nel listener degli eventi. In questo modo tutte le modifiche apportate al progetto da un guest ro vengono ignorate e quindi non producono alcun effetto.

4.2.2 Implementazione degli hint visuali

Gli hint visuali sono degli effetti grafici che informano visivamente l'utente di un'operazione eseguita nell'interfaccia dell'IDE da uno degli altri utenti della sessione, come per esempio la modifica alla proprietà di un componente oppure l'aggiunta di un nuovo componente all'interno del progetto.
Per la realizzazione di questa funzionalità viene utilizzato il type "view". Un client che effettua una modifica invia un messaggio che ha come type la stringa "view" e come content l'elemento DOM oggetto della modifica. Alla ricezione del messaggio, gli altri client chiamano il metodo createRipple() passandogli come parametro l'elemento DOM ricevuto. Il metodo utilizza le coordinate relative al posizionamento dell'elemento nella pagina per creare un div al centro dell'elemento stesso, sovrapposto a quest'ultimo. Il div viene stilizzato e animato in modo da realizzare l'effetto "ripple" che consiste in un cerchio rosso di dimensione 10 pixel che si allarga e si opacizza progressivamente fino a diventare invisibile. Ho realizzato questo effetto sfruttando le animazioni CSS3 definendo nel foglio di stile le seguenti regole:

```
@-Webkit-keyframes ripple{
 0% {opacity: 1;}
 100% {opacity: 0; transform: scale(2);}
}

.animate{
 -Webkit-animation: ripple 0.2s;
}
```

Ho settato infine un listener sull'evento onAnimationEnd in modo da rimuovere il div non appena l'animazione termina.

4.2.3 Implementazione della videochat

Come detto in precedenza, la possibilità di collaborare in videochat è prevista solo nel caso in cui alla sessione partecipino esattamente due utenti. Ho realizzato un protocollo che, attraverso uno scambio di messaggi, permette ai due utenti di iniziare una videochiamata. Analizziamo i passi che portano all'instaurazione del collegamento sia dalla parte del chiamante che dalla parte del chiamato.
La videochiamata viene avviata da uno dei due utenti cliccando il pulsante di chiamata posizionato sulla barra del titolo della chat. Il chiamante crea il proprio oggetto Peer e si mette in ascolto dell'evento on("open") del Peer. Lo scattare di questo evento indica che il peer è pronto alla connessione. Il chiamante invia allora un messaggio di tipo remoteIdReq per richiedere al peer remoto l'invio del proprio id e crea il tag video nel quale verranno inseriti gli stream locali. Successivamente chiama il metodo getMediaStream per recuperare i propri stream audio e video. Questo metodo fa uso dell'API getUserMedia nativa di WebRTC la cui chiamata fa apparire un popup permettendo all'utente di concedere o negare l'accesso alla Webcam e al microfono del dispositivo.
Il chiamato, quando riceve il messaggio di tipo "remoteIdReq", effettua le seguenti operazioni: crea l'oggetto Peer, crea il tag video in cui saranno caricati gli stream remoti e setta i listener sugli eventi on("open") e on("call") dell'oggetto Peer. Quando scatta l'evento on("open"), come avviene per il chiamante, viene creato il tag video per gli stream locali e viene chiamato il metodo getMediaStream. Una volta ottenuto l'accesso agli stream locali, questi vengono caricati nel tag video dedicato e viene inviato al chiamante un messaggio di tipo "calleeReady" contenete l'id del chiamato. Il messaggio viene ricevuto dal chiamante il quale crea il video remoto e setta la proprietà remoteId con il valore ricevuto.

Quando anche il chiamante concede l'accesso alla Webcam e al microfono viene chiamato il metodo call dell'oggetto Peer, passandogli come parametri la proprietà remoteId e gli stream locali. Questo metodo restituisce un oggetto MediaConnection che rappresenta la chiamata. Tale oggetto viene utilizzato per mettersi in ascolto sull'evento on("stream") in attesa dell'invio degli stream da parte del peer remoto. La chiamata del metodo call fa scattare l'evento on("call") su cui il chiamato si era messo precedentemente in ascolto. Al verificarsi di questo evento il chiamato risponde attraverso il metodo answer() dell'oggetto MediaConnection che rappresenta la chiamata in entrata e si mette in attesa degli stream remoti sull'evento on("stream"). Quando questi vengono ricevuti, vengono caricati nei tag video a loro dedicati e la videochat può avere inizio.

4.2.4 Termine della collaborazione

Il termine della collaborazione è fissato dall'owner attraverso la chiusura della chat di sessione. Il click sul pulsante di chiusura della chat comporta la deallocazione dell'oggetto ChatView e l'invio di un messaggio con type "disconnectClient". Quando i guest ricevono il messaggio effettuano un redirect all'URL http://www.instantdeveloper.com che comporta la loro esclusione dalla sessione.
Se invece il pulsante di chiusura viene cliccato da un guest, questo invia un messaggio con type "closeChat" e successivamente esce dalla sessione. Il messaggio inviato viene ignorato dagli altri guest poichè è indirizzato all'owner. Alla ricezione l'owner effettua un controllo sul numero dei guest partecipanti alla sessione e nel caso in cui il guest che si sta disconnettendo sia l'ultimo chiude la chat.

4.3 Componente WebRTC per le applicazioni

Il componente consente agli sviluppatori di integrare WebRTC nelle applicazioni Web sviluppate attraverso InDe RT rendendole in grado di scambiare

stream audio e video con altri software che supportino questa tecnologia.
Attraverso l'interfaccia dell'IDE è possibile inserirlo all'interno della propria applicazione. È costituito da un elemento div principale, che serve per visualizzare il video dell'utente remoto, e da un div interno a questo di dimensioni minori nel quale sarà visualizzato il video locale. Il risultato è una struttura compatta in cui il contenitore del video locale sarà sovrapposto a quello del video remoto.
Una volta integrato dentro l'applicazione è possibile programmare il componente grazie a proprietà, metodi ed eventi forniti allo sviluppatore come libreria.
All'interno del componente WebRTC viene creato un oggetto Peer che realizza tutta l'infrastruttura di basso livello per permettere la comunicazione WebRTC.
Nel pannello di destra dell'IDE sono visibili le informazioni relative al componente selezionato ed è da qui che si possono impostare le relative proprietà.
Le proprietà del componente WebRTC sono:

- peerId: è l'identificativo univoco del peer restituito dall'evento on("open") dell'oggetto Peer nel momento in cui il peer apre la sua connessione con il server. Lo si può pensare come un numero di telefono. Deve essere quindi condiviso con l'applicazione che ha intenzione di videochiamare il peer.
  Oltre che essere utilizzata per permettere la ricezione di videochiamate, la proprietà peerId è necessaria anche per avviarne una. In questo caso a questa proprietà deve essere assegnato l'id del peer che si vuole chiamare. Infatti il componente utilizzerà l'informazione contenuta in questa proprietà per avviare la videochiamata.
  Riassumendo, la proprietà peerId ha due funzioni: nel caso di ricezione di una chiamata serve per permettere al peer di essere identificato dal peer remoto; nel caso di avvio di una chiamata serve a consetirgli di identificare il peer remoto.

- sendAudio: è un valore booleano che specifica se all'avvio di una vi-

deochiamata verrà inviato lo stream audio . Se non settato il suo valore è true. Può essere modificato a chiamata in corso per spegnere o accendere il microfono;

- sendVideo: è un valore booleano che specifica se all'avvio di una videochiamata verrà inviato lo stream video. Se non settato il suo valore è true. Può essere modificato a chiamata in corso per attivare o disattivare la Webcam;

- localVideo: permette di stilizzare e posizionare il contenitore dentro al quale verrà visualizzato lo stream video catturato dalla propria Webcam. Deve essere impostato in formato JSON attraverso le proprietà width, position e style.

  – La proprietà width rappresenta la dimensione in percentuale del contenitore del video locale rispetto al contenitore del video remoto. Nel caso in cui non venisse impostata il suo valore di default è 25%.

  – La proprietà position permette di assegnare la posizione al video locale rispetto al video remoto. Ho previsto nove punti di posizionamento ottenibili attraverso la combinazione di una coordinata orizzontale, che può assumere i valori left, center e right, e di una coordinata verticale, che può assumere i valori top, center e bottom. Il suo valore di default è left top, quindi se non impostata il video locale si troverà nell'angolo in alto a sinistra del contenitore del video remoto.

  – La proprietà style è un oggetto JSON da settare utilizzando il formato delle regole CSS. Se non viene definita, il contenitore del video locale risulterà non stilizzato.
  Per chiarire il modo in cui può essere utilizzata la proprietà localVideo, vediamo un esempio in cui si vuole far sì che il video locale abbia una dimensione pari al 30% del video remoto, sia posizionato nell'angolo in basso a destra rispetto a quest'ultimo e abbia un

bordo arrotondato di colore rosso. In tal caso si dovrà scrivere:

```
{
   "width" : "30%",
   "position" : "right bottom",
   "style" : {
               "border" : "1px solid red",
               "border-radius" : "10%"
   }
}
```

- autoRespond: è un valore booleano che permette di decidere se una chiamata in entrata necessita di una risposta esplicita da parte dell'utente oppure se la risposta avviene in modo automatico. Se non impostata, il suo valore di default è false.

I metodi base del componente sono:

- startCall: permette di avviare una videochiamata o di rispondere ad una videochiamata in entrata. Viene cioè utilizzato sia per chiamare che per rispondere in base al valore della proprietà incomingCall interna al componente. Questa proprietà viene impostata nell'evento on("call") dell'oggetto Peer che viene scatenato quando c'è una chiamata in entrata e rappresenta un oggetto di tipo MediaConnection. Quindi nel caso di avvio di una chiamata questa proprietà risulterà non definita, mentre nel caso di ricezione di una chiamata rappresenterà la chiamata in entrata permettendo di effettuare una gestione diversa per i due casi. L'esecuzione del metodo startCall segue i seguenti passi:
  - chiede all'utente il consenso ad accedere alla Webcam e al microfono e in caso positivo accede agli stream audio e video utilizzando la funzione getUserMedia, che fa parte delle API di WebRTC;

  - setta il video locale con gli stream audio e video appena recuperati;
  - effettua la chiamata o la risposta in base alla proprietà incomingCall del componente WebRTC.
    * Se questa risulta undefined effettua la chiamata utilizzando il metodo call dell'oggetto Peer. Questo metodo prende come parametri gli stream audio e video locali e il valore della proprietà peerId che, come detto in precedenza, deve essere settato all'id del peer remoto da chiamare e restituisce un oggetto di tipo MediaConnection che rappresenta la chiamata.
    * Se invece è presente una chiamata in entrata viene effettuata la risposta attraverso il metodo answer chiamato sulla proprietà incomingCall. A questo metodo vengono passati gli stream locali come parametro.
  - si mette in ascolto, attraverso il listener on("stream") dell'oggetto MediaConnection, dell'invio degli stream audio e video da parte del peer remoto.
  - carica gli stream ricevuti dal peer remoto nel div dedicato.

- endCall: permette di terminare la videochiamata. Viene revocato l'accesso alla Webcam e al microfono chiamando il metodo stop sugli stream audio e video e viene chiamato il metodo close sull'oggetto incomingCall di tipo MediaConnection. In questo modo vengono bloccati rispettivamente l'invio degli stream locali e la ricezione degli stream remoti;

- openMultiClientPopup: crea una finestra per la videochat sottoforma di popup. Viene utilizzato nell'implementazione della videochat nelle sessioni multiutente a livello di applicazione.

Ho realizzato infine degli eventi programmabili che possono essere aggiunti al componente in fase di sviluppo. Questi eventi sono:

- onChange: scatta nel momento in cui cambia la proprietà peerId cioè quando il server assegna un id al peer. Aggiungendo questo evento al componente WebRTC si può rendere disponibile all'utente l'id del peer semplicemente accedendo alla proprietà peerId. Per esempio, si può assegnare il valore di questa proprietà ad una casella di testo in modo da permettere all'utente di copiarlo e condividerlo con un altro con cui vuole effettuare una videochiamata;

- onIncomingCall: scatta nel momento in cui il peer riceve una chiamata. Può essere utilizzato per realizzare la gestione della risposta prevedendo per esempio un popup che permetta di accettare o rifiutare la chiamata. Nel caso in cui la proprietà autoRespond sia stata settata a false, qui deve essere chiamato il metodo startCall() per rispondere alla chiamata in entrata;

- onStartCall: scatta nel momento in cui il peer riceve gli stream remoti e quindi la chiamata è iniziata;

- onEndCall: scatta alla chiusura della chiamata;

- onError: scatta al verificarsi di un errore, ovvero quando il peer non riesce a collegarsi al server per ottenere l'id oppure quando si verifica un errore in fase di avvio o durante una chiamata.

4.4 Sessioni collaborative a livello di applicazione

Passiamo ora a descrivere come avviene la collaborazione in una sessione di un'applicazione.
L'applicazione è implementata attraverso due componenti, la classe App lato server e la classe App lato client. L'applicazione in esecuzione su un dispositivo è un oggetto della classe App lato client. Il corrispondente oggetto lato server detiene il controllo sulla logica dell'applicazione. La comunicazione tra

App lato server e App lato client avviene attraverso una Websocket: l'oggetto App lato server invia dei messaggi che contengono dei comandi che vengono eseguiti sui client, l'oggetto App lato client invece invia eventi che rappresentano operazioni effettuate sull'applicazione dall'utente. Queste operazioni vengono eseguite sull'App a livello server e propagate lato client attraverso l'invio di comandi.
All'esecuzione di un'applicazione è associata una sessione. Come avviene a livello di IDE, per avviare una collaborazione è necessario condividere la sessione. A tale scopo ho realizzato il metodo requestConnectionURL della classe App lato server. Questo metodo restituisce un URL con una query string formata da un parametro sid che rappresenta l'id della sessione, un parametro appid che rappresenta l'applicazione lato server e un parametro acid che rappresenta il nuovo client. Il parametro acid viene memorizzato dal server associandolo al sid e all'appid. Quando un client accede all'URL, il server effettua un controllo sui tre parametri per verificare se l'acid fornito è associato al sid e all'appid. In caso affermativo invia il comando di esecuzione dell'applicazione sul client collegato, ammettendolo alla sessione.
Nei paragrafi successivi esporrò le funzionalità previste nella collaborazione a livello di applicazione.

4.4.1 Gestione dei ruoli e implementazione della videochat

Come detto in precedenza, la videochat è una caratteristica di base della sessione multiutente. Questo vuol dire che quando un client entra a far parte della sessione viene avviato in automatico il meccanismo di creazione della videochiamata. All'interno di una sessione collaborativa multiutente i client hanno un ruolo che viene assegnato loro dall'owner della sessione.
Per consentire all'owner di assegnare il ruolo al client che si connette alla sessione e di settare i parametri relativi alla videochiamata ho realizzato l'evento onConnectingClient della classe App lato server. Questo evento

è programmabile in fase di sviluppo dell'applicazione e ha un parametro settings costituito da un oggetto con le seguenti proprietà:

- clientRole: permette di specificare il ruolo del client all'interno della sessione. Può assumere i valori:
  - "ro", per assegnare al guest il ruolo readonly. Il ruolo readonly toglie la possibilità di interagire con l'interfaccia al guest a cui è assegnato: infatti gli eventi relativi ad operazioni che questo esegue non vengono inviati al server e quindi non producono alcun effetto;
  - "rw", per assegnare al guest il ruolo readwrite. Il ruolo readwrite consente al guest di essere operativo rendendolo in grado di apportare modifiche all'applicazione.

  Se non settato il suo valore di default è "ro";

- ownerStream: permette di specificare quali stream l'owner della sessione invierà ai guest. Può essere impostato ad "a" per inviare il solo stream audio, a "v" per inviare solo lo stream video oppure ad "av" per inviare entrambi gli stream. Ho deciso di impostare il suo valore di default ad "a" per lasciare all'owner la possibilità di decidere se inviare o meno il proprio stream video;

- guestStream: permette di specificare gli stream che il guest invierà all'owner. Il suo settaggio avviene allo stesso modo della proprietà ownerStream. Il suo valore di default è "av". Ho effettuato questa scelta nell'ottica di un caso d'uso legato all'assistenza remota: l'utente che richiede l'assistenza remota vedrà e sentirà l'assistente, mentre quest'ultimo potrà solo sentire l'utente il quale potrà decidere in un secondo momento se condividere il proprio video;

- ownerPeerId: rappresenta l'id dell'oggetto Peer creato dall'owner.

Queste proprietà sono tutte accessibili in fase di programmazione dell'evento, ad eccezione della proprietà ownerPeerId che serve per permettere ai guest di collegarsi in videochat con l'owner e quindi non deve essere modificata. Vediamo ora in dettaglio come funziona l'evento onConnectingClient.
Il funzionamento dell'evento onConnectingClient è il seguente: nel momento in cui un client si connette, invia all'oggetto App lato server un messaggio che notifica il suo ingresso nella sessione. La ricezione di questo messaggio lato server fa scattare l'evento. L'oggetto App lato server invia ai client un messaggio contenente l'oggetto settings. Alla ricezione del messaggio, i client chiameranno il metodo setClientProperties passandogli come parametro l'oggetto settings ricevuto. Questo parametro viene utilizzato in modo diverso a seconda che il client che chiama il metodo sia l'owner o un guest. Infatti l'owner userà solo il campo ownerStream per sapere quali stream dovrà inviare all'avvio della videochiamata. I guest useranno tutti i rimanenti: il campo clientRole per settare il proprio ruolo, il campo guestStream per sapere quali stream inviare e il campo ownerPeerId per avviare una videochiamata con l'owner.

Avvio della videochiamata
Il metodo setClientProperties, oltre a settare le opzioni relative al collegamento dei client, si occupa anche di creare la finestra per la videochat e di avviare la videochiamata. Per fare ciò si avvale della classe WebRTC utilizzandola in maniera leggermente diversa da quella tradizionale. La sessione di collaborazione potrebbe essere relativa ad un'applicazione che non prevede l'utilizzo di WebRTC tra le sue funzionalità e quindi non dispone di un componente WebRTC nella sua struttura. Ho voluto quindi creare la videochat sottoforma di popup, in modo che al termine della sessione il popup possa essere rimosso riportando l'applicazione allo stato originale e senza aver apportato modifiche alla struttura dei suoi componenti.
Come detto precedentemente, è il client guest ad avviare la videochiamata. I passi che portano all'instaurazione del collegamento sono i seguenti:

- sia l'owner che il guest creano un oggetto della classe WebRTC e chiamano il metodo openMultiClientPopup di questa classe. Questo metodo crea la finestra della videochat e i controlli per la videochiamata: un pulsante per accendere e spegnere la videocamera, un pulsante per attivare e disattivare il microfono e un pulsante per chiudere il collegamento;
- il guest utilizza il campo ownerPeerId dell'oggetto settings ricevuto dal server per settare la proprietà peerId del proprio oggetto WebRTC;
- il guest chiama il metodo startCall della classe WebRTC per avviare la chiamata.

La creazione della videochiamata avviene quindi seguendo il protocollo di comunicazione implementato nella classe WebRTC.

4.4.2 Implementazione del data masking

Nella fasi di sviluppo di un'applicazione all'interno di InDe RT tutti i componenti che rappresentano elementi DOM sono implementati come classi che estendono la classe generica Element. Per implementare il data masking ho aggiunto a questa classe la proprietà ds che contrassegna come dato sensibile il componente su cui è definita. Dopo aver creato un componente di un'applicazione all'interno dell'IDE la si può trovare come proprietà booleana nel pannello in cui sono elencate tutte le proprietà editabili del componente. Durante una sessione di collaborazione l'aspetto grafico dei componenti la cui proprietà ds sia stata settata a true verrà modificato per i client guest in modo da nascondere il dato che il componente rappresenta. Il mascheramento riguarda quattro proprietà che i componenti possono avere:

- src;
- innerHTML;
- value;

- innerText;

Proprietà src

Gli oggetti su cui la proprietà src risulta definita sono le immagini. Il mascheramento delle immagini avviene applicando una regola CSS che applica un filtro blur sull'immagine.

Proprietà innerHTML

Il mascheramento avviene sostituendo il contenuto di questa proprietà con una stringa di asterischi di lunghezza prefissata.

Proprietà value e innerText

Il valore di queste due proprietà è costituito da stringhe. Le stringhe vengono mascherate sostituendo i caratteri che le compongono con degli asterischi. Se la lunghezza della stringa è superiore a 50 caratteri il mascheramento viene completato aggiungendo dei puntini sospensivi evitando di rappresentare la stringa nella sua interezza.

4.4.3 Termine della collaborazione

In una sessione multiutente di un'applicazione la videochat rappresenta una funzionalità di base della collaborazione. Ecco perchè ho deciso di associare il termine della collaborazione alla chiusura della videochiamata. Questa avviene quando l'owner clicca sul pulsante di chiusura presente tra i controlli della propria finestra di videochat. Al click su questo pulsante avvengono le seguenti operazioni:

- viene chiusa la videochiamata e distrutto il popup della videochat;
- viene inviato un messaggio di disconnessione dei guest all'oggetto App lato server. Questo inoltra il messaggio a tutti i client della sessione. Il client owner ignorerà il messaggio. I guest invece effettueranno un redirect all'indirizzo http://www.instantdeveloper.com uscendo dalla sessione.

Un guest può anche uscire dalla sessione di collaborazione autonomamente cliccando sul pulsante di chiusura della videochiamata. Questo comporta il suo reindirizzamento all'URL sopra citato e quindi la sua disconnessione dalla sessione, ma non inficia il proseguimento della collaborazione da parte dei client rimanenti.

Capitolo 5

Valutazione

Per poter utilizzare in un ambiente reale un sistema di collaborazione real-time occorre che esso raggiunga un certo livello di efficienza e affidabilità per garantire che i partecipanti non percepiscano rallentamenti o blocchi e siano sempre sincronizzati. Per verificare se il sistema di collaborazione realizzato potesse davvero essere usato in situazioni reali, ho realizzato una serie di test prendendo in considerazione diversi scenari di utilizzo. Nei paragrafi successivi vedremo gli strumenti utilizzati per realizzare i test, i casi di utilizzo del sistema, i parametri presi in considerazione per valutarlo e il risultato della valutazione.

5.1 Strumenti

I test sono stati realizzati utilizzando sessioni collaborative dell'IDE sul browser Google Chrome in esecuzione su macchine con sistema operativo Windows 8.1. L'IDE è in esecuzione su un server dedicato che utilizza Node.js. Le comunicazioni tra client e server avvengono attraverso Websocket. Per quanto riguarda l'aspetto WebRTC, sempre lo stesso server agisce anche da server STUN e, se necessario, da server TURN per permettere le videochiamate nel caso in cui il collegamento peer-to-peer non sia possibile. I

dati e le statistiche relativi alle videochiamate sono stati ottenuti utilizzando l'API di WebRTC getStats.

5.2 Scenari di utilizzo

L'aspetto più interessante nella valutazione del sistema riguarda l'incidenza di WebRTC sulle performance generali della collaborazione. Per questo motivo ho effettuato dei test per analizzare lo stato del sistema durante un collegamento in videochat.

In particolare ho valutato le performance della collaborazione nei seguenti scenari:

- collaborazione con collegamento in videochiamata one-to-one;
- collaborazione con collegamento in videochiamata many-to-many.

Nel caso many-to-many la videochiamata è stata testata con lo scambio dei soli stream audio tra i peer.

5.3 Parametri di valutazione

I parametri utilizzati nella valutazione sono di due tipi: parametri di network e parametri di host.

5.3.1 Parametri di network

Per quanto riguarda le connessioni sulla rete sono stati considerati tre fattori globali: la perdita di pacchetti, la banda occupata e il ritardo.

Perdita di pacchetti
Il tasso di perdita indica le perdite di pacchetti durante la trasmissione. Di solito la perdita di pacchetti condiziona direttamente la qualità di una chiamata. Nel caso di WebRTC la perdita di pacchetti è un indicatore della

qualità della trasmissione; infatti in caso di una grossa perdita di pacchetti lo stream potrebbe risultare degradato compromettendo la qualità del video. Le perdite sono calcolate in un certo periodo di tempo, utilizzando l'equazione seguente attraverso la quale è possibile calcolare il tasso di perdita su una certa connessione:

$$\frac{PKT_{loss}(T) - PKT_{loss}(T-1)}{PKT_{received}(T) - PKT_{received}(T-1) + PKT_{loss}(T) - PKT_{loss}(T-1)}$$

Questa equazione è stata applicata periodicamente per ottenere il tasso di perdita.

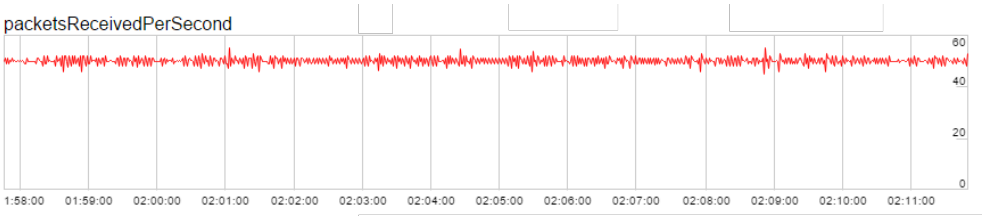

Figura 5.1: Pacchetti audio ricevuti

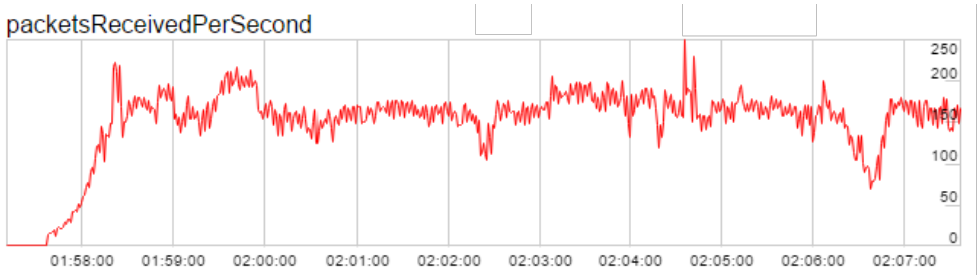

Figura 5.2: Pacchetti video ricevuti

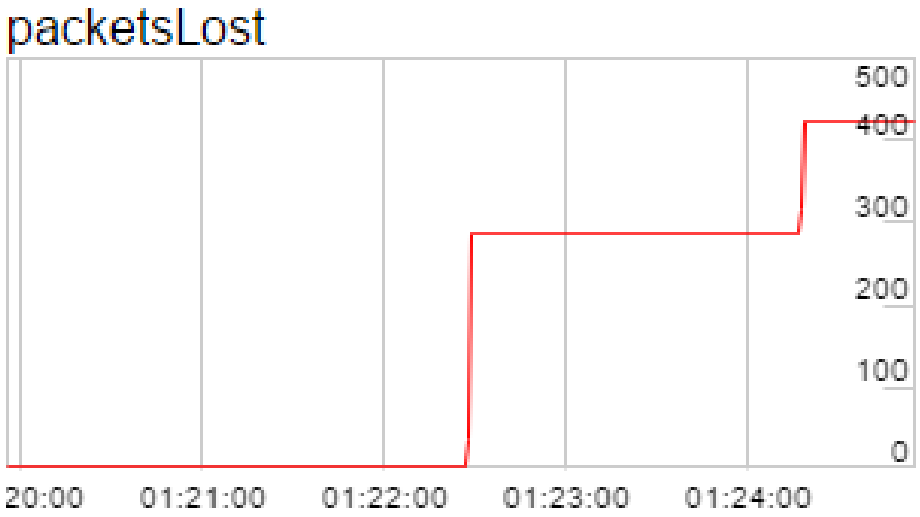

Figura 5.3: Pacchetti persi

Figura 5.4: Tasso di perdita per unità di tempo

Throughput

Il throughput è un parametro chiave per misuare le performance delle applicazioni che utilizzano WebRTC. Questo parametro indica la capacità di trasmissione effettivamente utilizzata di un canale di comunicazione e può essere diviso in tre componenti:

- sender rate: numero di pacchetti inviati per unità di tempo;

- receiver rate: numero di pacchetti ricevuti per unità di tempo;

- goodput: quantità di dati utili in un'unità di tempo.

Per calcolare il throughput mi sono basato sui risultati forniti da getStats. In particolare ho preso in considerazione il numero di byte ricevuti in un certo periodo di tempo utilizzando la seguente formula: $\frac{\Delta Bytes_{received}}{\Delta time}$.
Il throughput definisce l'utilizzo di banda da parte degli stream audio e video e può essere utilizzato per valutare la qualità della chiamata. Più il suo valore è basso, più la banda disponibile è ridotta con possibili problemi di trasmissione o, nei casi peggiori, perdita di comunicazione tra i peer.

Figura 5.5: Throughput per stream audio in connessione one-to-one

Figura 5.6: Throughput per stream video in connessione one-to-one

Come si può notare dai grafici, lo stream video ha un throughput medio di circa 1 MegaByte. Questo vuol dire che ogni peer collegato caricherà la banda con 1 MegaByte aggiuntivo. Risulta evidente come la videochiamata multipla non scali in quanto basta un numero non elevato di utenti collegati, ad esempio tre o quattro, per appesantire una normale connessione ADSL causando rallentamenti e blocchi. Da qui la scelta di implementare le videochiamate multiple mediante l'invio del solo stream audio che invece ha un throughput medio di circa 40 KiloByte.

Round-Trip Time (RTT) e One-Way Delay (OWD)
Il ritardo sulla connessione può essere misurato in due modi diversi:

- Round-Trip Time (RTT): indica il tempo impiegato da un pacchetto per viaggiare dal mittente al destinatario e tornare indietro. Il calcolo di questo parametro viene effettuato dall'API getStats di WebRTC.

- One-Way Delay (OWD): indica il tempo impiegato da un pacchetto per spostarsi da un peer ad un altro. Questo tempo include diversi ritardi prodotti lungo la connessione. OWD è calcolato sommando:

Utenti in videochat	RTT medio
2	$43Ms$
5	$64Ms$
10	$106Ms$

Tabella 5.1: Round-Trip Time medio

- transmission delay: il tempo richiesto per immettere il pacchetto nel canale;
- propagation delay: il tempo che il pacchetto trascorre sul canale;
- processing delay: il tempo che i router in entrata e in uscita impiegano per processare il pacchetto;
- queuing delay: il tempo che il pacchetto trascorre in coda in attesa di essere processato dal router.

$OWD = delay_{transmission} + delay_{propagation} + delay_{processing} + delay_{queuing}$
Il calcolo del OWD non è un'operazione semplice e richiede che i clock di entrambe le macchine siano accuratamente sincronizzati. Per i miei test ho assunto che il ritardo fosse simmetrico e ho calcolato l'OWD con la formula $\frac{RTT}{2}$.

5.3.2 Parametri di host

I parametri di host sono delle misure effettuate localmente all'host che riguardano il comportamento delle comunicazioni real-time. Questi parametri non sono necessariamente legati alla rete, ma danno informazioni sulle performance dell'host quando utilizza WebRTC.

Risorse
Una delle risorse critiche nell'utlizzo di WebRTC è la CPU. Il carico della CPU è determinato dal numero di stream da gestire. Un carico normale aiuta

WebRTC ad avere migliori performance. Un'altra risorsa presa in considerazione è la RAM. La quantità di RAM utilizzata durante un collegamento WebRTC è importante soprattutto per i dispositivi mobili. Ho osservato l'utilizzo di questa risorsa per calcolare lo stress dell'host durante l'utilizzo di WebRTC.

Setup time
Il setup time è il tempo necessario per instaurare il collegamento ed è stato calcolato utilizzando dei timestamp inseriti in due punti specifici: nel momento della chiamata e nel listener dell'evento peer.on("call"). Sottraendo questi due tempi ho ottenuto il valore del parametro di setup.

Call failure
Il call failure è un parametro che misura la qualità del software di un'applicazione che fa uso di WebRTC e indica il tasso di successo nello stabilire una chiamata WebRTC.

Utenti	**Utilizzo CPU**	**RAM occupata**	**Setup time**	**Call failure**
2	+7%	+0.08GB	54Ms	0%
5	+34%	+0.27GB	92Ms	0%
10	+61%	+0.52GB	113Ms	0%

Tabella 5.2: Parametri di host per videochiamate con 2, 5 e 10 partecipanti. I dati relativi alla CPU e alla RAM indicano una maggiorazione rispetto allo stato precedente: per la riga relativa al caso di 2 utenti lo stato precedente è rappresentato dai valori esistenti prima dell'instaurazione del collegamento WebRTC

5.4 Risultati

Dalle analisi e dai test effettuati sul sistema di collaborazione realizzato è emerso che la tecnologia WebRTC è molto performante in quanto ad affidabilità e a possibilità di collaborare real-time nel caso di collegamenti one-to-one, mentre richiede alcuni accorgimenti per consentire la collaborazione multiutente. È infatti necessario disabilitare lo scambio degli stream video se si vogliono mantenere gli stessi livelli di efficienza anche nelle videochiamate multiclient. Comunque oltre un certo limite di collegamenti contemporanei, anche adottando questo accorgimento i risultati della collaborazione non sono più accettabili. Nei test da me effettuati ho potuto notare come questo limite non possa oltrepassare le 10 unità. Tuttavia, durante una collaborazione che abbia come scopo lo sviluppo software un numero così elevato di utenti collegati è piuttosto improbabile perchè rischierebbe di compromettere la buona riuscita della collaborazione stessa anche a livello di interazione tra gli utenti. Pertanto le performance del sistema di collaborazione implementato rispondono ai requisiti funzionali richiesti e sono da considerarsi soddisfacenti.

Capitolo 6

Conclusioni

In questa tesi è stato affrontato il problema della collaborazione in tempo reale attraverso il Web in un ambiente di sviluppo software browser-based.
Per prima cosa sono state introdotte le diverse forme di collaborazione via Web e gli strumenti che le rendono possibili. È stato analizzato come la collaborazione in tempo reale possa applicarsi a diversi ambiti e possa essere utilizzata con finalità differenti, a partire dalla semplice videochat fino ad arrivare a contesti più complessi come per esempio la collaborazione nell'ambito dello sviluppo di progetti software. In particolare è stato approfondito quest'ultimo aspetto e introdotto il lavoro oggetto di questa tesi: la realizzazione di un sistema di collaborazione real-time all'interno di un IDE browser-based, InDe RT.
InDe RT permette lo sviluppo di applicazioni mobile e Web e la sua struttura si prestava a farlo diventare un IDE collaborativo. È stata quindi presentata l'implementazione per la gestione della collaborazione in due contesti distinti. Il primo riguarda la collaborazione a livello di IDE e consiste in una serie di funzionalità che permettono a due o più utenti di poter sviluppare uno stesso progetto software contemporaneamente, condividendo lo stesso spazio di lavoro, i dati e le informazioni all'interno di esso. Il secondo offre la possibilità per più utenti di utilizzare un'applicazione condividendo l'interfaccia utente e i dati in essa contenuti.

In seguito si è descritta l'integrazione di servizi aggiuntivi a supporto della collaborazione. A questo scopo è stato introdotto WebRTC, una tecnologia che permette lo scambio di stream audio e video direttamente tra browser. All'interno del sistema di collaborazione questa tecnologia è stata utilizzata in tre forme:

- videochiamate nella collaborazione a livello IDE;
- videochiamate nella collaborazione a livello di applicazioni;
- componente per lo sviluppo di applicazioni dotate del supporto WebRTC.

È stata poi presentata la funzionalità legata alla gestione dei ruoli degli utenti durante una collaborazione, attraverso la quale il proprietario della sessione può esercitare un controllo sulla sessione e sui suoi partecipanti.
Successivamente si è evidenziato come particolare attenzione sia stata riservata alla privacy dell'utente durante le sessioni di collaborazione. È stato spiegato a questo proposito come la realizzazione del sistema di data masking permette di nascondere i dati che l'utente proprietario della sessione considera sensibili.
Infine sono state analizzate le performance del sistema sviluppato in termini di banda utilizzata e affidabilità.

Per quanto riguarda alcuni dei possibili sviluppi futuri uno di questi consiste nella possibilità di effettuare videochiamate complete, comprensive di stream video, durante le sessioni che coinvolgono più di due utenti. Al momento gli stream video possono essere condivisi soltanto quando la collaborazione coinvolge esattamente due utenti. Per le videochiamate con più di due partecipanti è permesso solo lo scambio dello stream audio in quanto lo scambio di molti stream video saturerebbe la banda. Questa limitazione potrebbe essere superata con l'utilizzo di un server dedicato che si occupi di effettuare il multiplexing degli stream video.
Un altro avanzamento del sistema di collaborazione potrebbe consistere nel

renderlo disponibile sui principali browser. Il software all'interno del quale questo progetto è stato inserito è pensato per essere utilizzato principalmente sul browser Google Chrome. Per questo motivo il sistema attuale è stato testato soltanto su Google Chrome. Dal momento che WebRTC è uno standard supportato anche da Mozilla Firefox e Opera sarebbe utile assicurare che le funzionalità implementate siano disponibili anche su questi browser. In questo modo sarebbe possibile realizzare delle sessioni di collaborazione i cui partecipanti operino su browser diversi.

Bibliografia

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] G. Allen, F. Loffler, T. Radke, E. Schnetter, and E. Seidel, "Integrating web 2.0 technologies with scientific simulation codes for real-time collaboration," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pp. 1–10, IEEE, 2009.

[3] E. Bertin, S. Cubaud, S. Tuffin, S. Cazeaux, N. Crespi, and V. Beltran, "Webrtc, the day after," 2013.

[4] S. K. Badam and N. Elmqvist, "Polychrome: A cross-device framework for collaborative web visualization," in *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, pp. 109–118, ACM, 2014.

[5] R. Calderon, M. Blackstock, R. Lea, S. Fels, A. de Oliveira Bueno, and J. Anacleto, "Red: a framework for prototyping multi-display applications using web technologies," in *Proceedings of The International Symposium on Pervasive Displays*, p. 148, ACM, 2014.

[6] J. Lautamäki, A. Nieminen, J. Koskinen, T. Aho, T. Mikkonen, and M. Englund, "Cored: browser-based collaborative real-time editor for

java web applications," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1307–1316, ACM, 2012.

[7] T. Aho, A. Ashraf, M. Englund, J. Katajamäki, J. Koskinen, J. Lautamäki, A. Nieminen, I. Porres, and I. Turunen, "Designing ide as a service," *Communications of Cloud Software*, vol. 1, no. 1, 2011.

[8] P. Nicolaescu, M. Derntl, and R. Klamma, "Browser-based collaborative modeling in near real-time," in *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, pp. 335–344, IEEE, 2013.

[9] J. Rosenberg and A. Keranen, "Interactive connectivity establishment (ice): A protocol for network address translator (nat) traversal for offer/answer protocols," 2013.

[10] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session traversal utilities for nat (stun)," tech. rep., RFC 5389 (Proposed Standard), 2008.

[11] R. Mahy, P. Matthews, and J. Rosenberg, "Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun)," *Internet Request for Comments*, 2010.