Studio di fattibilit‘a

Informazioni sul documento	
Nome file:	SF.tex
Versione:	1.4
Data creazione:	
Data ultima modifica:	
Stato:	
Uso:	
Redattori:	
Approvato da:	
Verificatori:	

Storia delle modifiche

Versione	Descrzione intervento	Redattore	Data
0.0	Creazione del documento e definizione dei punti chiave del documento. Stesura della "Descrizione sommaria del capitolato".	Andrea Rizzi	28/11/2012
1.0	Stesura del punto "Dominio tecnologico" (introduzione). Stesura del punto "Valutazione rischi". Stesura del punto "Vantaggi vari ed evventuali". Stesura del punto "Dominio applicativo".	Andrea Rizzi	29/11/2012
1.1	Stesura del punto "Java - webSoket".	Marco Schivo	29/11/2012
1.2	Stesura del punto "Javascript" e "Protocolli e funzionalità di Google Chrome".	Diego Beraldin	30/11/2012
1.3	Stesura del punto "WebRTC".	Andrea Rizzi	30/11/2012
1.4	Stesura del punto "HTML5".	Stefano Farronato	30/11/2012
1.5	Stesura del punto "CSS3".	Diego Beraldin	1/12/2012

Indice

Sommario

La presente, vuol'essere una bozza per un ipotetico studio di fattibilità. Di seguito verranno discussi i punti salienti che riguardano il capitolato C1, del corso di ingegneria del software. Si cercherà quindi di stabilire quali tecnologie sono necessarie al conseguimento dell'obbiettivo, e le problematiche insite nell'affrontarlo, sia sul piano dei requisiti che sul piano delle tecnologie.

1 Descrizione sommaria del capitolato

Il capitolato C1 (denominato MyTalk) proposto dall'azienda italiana Zucchetti, prevede la creazione di un software di comunicazione audio/video, basato sul progetto (attualmente ancora in fase di sviluppo) WebRTC. Proprio perchè nato di recente, WebRTC è facilemtne soggetto a modifiche (si pensi che in relazione alla data in cui è scritto il presente, l'ultima modifica risale al 15 novembre). Di conseguenza il committente ha precisato i seguenti punti fondamentali:

- l'architettura software deve basarsi su un modello elastico e facilmente scalabile in seguito a modifiche del pacchetto WebRTC;
- i requisiti opzionali sono modificabili/eliminabili/aggiungibili in corso d'opera, proprio perchè a priori non è evidentemente chiara la loro fattibilità.

Senza prendere in considerazione la completezza dei requisiti obbligatori, riportiamo di seguito alcuni punti fondamentali da cui è possibile trarre spunti per la determinazione di problemi tecnici e progettuali (che per l'appunto sono lo scopo di tale documento):

- l'applicativo non dovrà richiedere installazione ne di plugin ne di componenti aggiuntivi. Sfrutterà semplicemente il browser Chrome;
- di base, vi è un server scritto in java con l'implementazione di webSocket, al quale i client dovranno connettersi, ma che non dovrà partecipare alla comunicazione tra gli utenti;
- comunicazione audio;
- comunicazione video,

2 Studio del dominio

2.1 Dominio tecnologico

La realizzazione del progetto MyTalk richiede la conoscenza di alcunistrumenti tecnologici obbligatori, senza i quali sarebbe impossibile rispondere alle esigenze del committente. Di seguito riportiamo un elenco delle conoscenze richieste, affiancate da:

- una breve descriione;
- una serie di riferimenti alle parti che potrebbero (sempre in termini di anlisi preliminare) richiedere una conoscenza più o meno elevata di tale tecnica/tecnologia;
- una serie di link a guide o dispense informative, per una maggior trattazione dell'argomento.

2.1.1 WebRTC

Descrizione: WebRTC è un progetto open source di comunicazione audio/video real time proposto da Google. Il progetto consiste in una serie di API scritte in javascript. Ad oggi (data di stesura del documento), WebRTC è ancora in fase di sviluppo, o meglio, non ha ancora raggiunto una versione stabile e convalidata dal W3C. Stando agli sviluppatori del progetto, WebRTC dovrà fornire la possibilità di creare applicazioni web per il realtime multimedia (ad esempio: video chat), senza richiedere l'installazione di plugin o contenuti aggiuntivi.

In ambito più generale, l'opera di standardizzazione della comunicazione real time mediante applicativi web, passa attraverso due aspetti: la standardizzazione delle interfacce

programmatiche che i browser offriranno agli sviluppatori di applicazioni, e i protocolli per lo scambio di media tra browser e browser. I due aspetti sono affrontati rispettivamente dalle iniziative WebRTC in w3c, e RTCWEB in IETF. In sintesi RTCWEB rappresenta l'aspetto di basso livello della comunicazione RTC in ambito web, mentre WebRTC rappresenta l'alto livello, ossia le interfacce (tra cui quella) per la gestione dei flussi audio e video.

Riferimenti: "Il progetto deve essere basato sulla tecnologia WebRTC, parte delle proposte di evoluzione dell'HTML5."

Link utili:
Sito ufficiale del progetto WebRTC:
http://www.webrtc.org/

Pagina con le specifiche delle API:
http://dev.w3.org/2011/webrtc/editor/webrtc.html

Per una lettura introduttiva, consultare l'allegato: "Introduzione WebRTC.pdf";

2.1.2 Java - webSoket

Descrizione: è un'implementazione java e javascript del protocollo WebSocket HTML5. Questa contiene jWebSocketServer che è la parte di nostro interesse: una implementazione basata su Java per soluzioni streaming bidirezionali server-to-client e comunicazione client-to-client. L'idea è che il server utilizzi questa tecnologia per restare in ascolto di richieste da parte dei client, in questo caso l'avvio di chiamate e videochiamate. Nel sito riportato all'inizio, c'è molta documentazione che spiega bene nei dettagli come avviene il tutto.

Riferimenti: dal capitolato: "La parte server, necessaria solo nella fase di inizializzazione della chiamata, dovrà essere realizzata in Java e utilizzare il protocollo di comunicazione WebSocket."

Link utili:
http://code.google.com/p/websockets4j/

http://technology.amis.nl/2012/01/14/implementing-a-java-server-side-component-for-jwebsocket-server-for-websocket-interaction-with-web-clients/

2.1.3 Javascript

Descrizione: JavaScript è un linguaggio di scripting interpretato, debolmente orientato agli oggetti la cui sintassi è derivata dal C e il cui modello a oggetti è radicalmente diverso da quello del C++/Java a noi noto da altri corsi: ad es. alle classi possono essere aggiunte dinamicamente proprietà e operazioni, non esiste ereditarietà di classe ma solo fra oggetti (mediante prototipi) né *dynamic dispatch*, non c'è *encapsulation* in quanto tutte le proprietà degli oggetti sono accessibili come in un array associativo né protezione delle informazioni con i consueti modificatori di accessibilità.

A dire il vero non esistono nemmeno delle vere e proprie classi: gli oggetti possono essere definiti aggiungendo nuove proprietà e metodi (oggetti di tipo funzione, vd. poi) a oggetti preesistenti (es. un'istanza `obj` di `Object`) oppure definendo una funzione 'costruttore' che restituisca un'istanza di oggetto dopo averla costruita (usando `this` invece di `obj`). Con alcuni accorgimenti è possibile simulare la presenza di variabili di istanza private

(es. usando variabili locali dentro i costruttori), l'ereditarietà, l'overriding di metodi e, in generale, i meccanismi tipici del paradigma OO basato su classi.

Il sistema dei tipi di dato comprende `Array`, `Boolean`, `Date`, `Function`, `Math`, `Number`, `Object`, `Regexp` e `String`, tuttavia JavaScript è debolmente tipizzato e, in particolare, le variabili sono dichiarate assegnando un valore a un identificatore eventualmente preceduto dalla keyword `var` senza però specificarne il tipo. Nota: come forse si è intuito il fatto che le funzioni siano oggetti permette di utilizzare anche funzioni 'anonime' (di fatto è un sistema comodo per definire metodi in un costruttore o per passare funzioni di *callback* a funzioni di ordine superiore). Come se ciò non fosse sufficiente, il linguaggio supporta anche costrutti simili alle *closure* di Smalltalk (e derivati, es. Ruby) per cui anche i blocchi di codice sono visti come espressioni e possono essere passati come parametro, restituiti, assegnati a identificatori ecc...

Poiché JavaScript è utilizzato in genere all'interno di un programma ospite (*host*), il sistema dei tipi può essere espanso dai nuovi tipi facenti parte dell'API esposta dall'*host* verso l'ambiente JavaScript. Se l'host è un web browser – come tipicamente accade sfruttando JavaScript per la programmazione web lato client – il DOM fa parte dell'API resa disponibile dal browser ai programmi JavaScript e tutti i suoi elementi saranno accessibili (a partire da `document`,`form`, `link`, ecc...).[1]

JavaScript è impiegato per la costruzione di pagine web dinamiche (avvantaggiato in questo dal fatto di svolgere la computazione lato client), compito semplificato dalla possibilità di gestire gli eventi impostando gli *handler* associati agli oggetti per i vari tipi di evento (l'handler impostato può essere un oggetto di tipo `Function`).

Integrare il codice JavaScript all'interno di una pagina web è possibile fra i tag `<script>`...`<script>` con l'attributo `type` per specificare `text/javascript`. Il codice dello script può essere inserito nell'elemento head o nel body all'interno della pagina o, se viene richiamato in più pagine, in un file di testo esterno con estensione `.js` incluso mediante l'attributo `src` in un elemento `script`.

Riferimenti: «il programma che sarà realizzato non deve essere inteso come una pagina Web ma come un software che per l'occasione utilizzi il linguaggio Javascript e le librerie contenute nel browser»

Link utili:
Alcune fonti da cui ho liberamente attinto (oltre Wikipedia):
www.w3schools.com/js/default.asp
www.html.it/guide/guida-javascript-di-base/

2.1.4 JQuery

Descrizione:

Riferimenti:

Link utili:

2.1.5 HTML5

Descrizione: l'HTML5 è un linguaggio di markup per la strutturazione delle pagine web che e si pose come obiettivo quello di progettare specifiche per lo sviluppo di applicazioni web, focalizzandosi su miglioramenti e aggiunte ad HTML e alle tecnologie correlate.

[1] Molto comune è recuperare il riferimento a un elemento identificato dall'attributo `id` nella struttura dell'HTML chiamando il metodo `document.getElementById(pippo)`.

Le novità introdotte dall'HTML5 rispetto all'HTML 4 sono finalizzate soprattutto a migliorare il disaccoppiamento tra struttura, definita dal markup, caratteristiche di resa (tipo di carattere, colori, eccetera), definite dalle direttive di stile, e contenuti di una pagina web, definiti dal testo vero e proprio. Inoltre l'HTML5 prevede il supporto per la memorizzazione locale di grosse quantità di dati scaricati dal browser, per consentire l'utilizzo di applicazioni basate su web (come per esempio le caselle di posta di Google o altri servizi analoghi) anche in assenza di collegamento a Internet.

In particolare nella quinta versione di HTML:
-regole più stringenti per la strutturazione del testo in capitoli/paragrafi/sezioni;
-introdotti elementi di controllo per i menu di navigazione;
-migliorati ed estesi gli elementi di controllo per i moduli elettronici;
-introdotti specifici elementi per il controllo di contenuti multimediali (aggiungento di tag <video> e <audio>);
-tolti alcuni elementi di scarso o nessun uso effettivo;
-estesi a TUTTI i tag una serie di attributi (specialmente legati all'accessibilità);
-supportato Canvas, che permette di utilizzare JavaScript per creare animazioni e grafica bitmap;
-introdotta la geolocalizzazione;
-sistema alternativo ai normali cookie (denominato Web Storage),con maggiore efficenza e risparmio di banda;
-standardizzazione di programmi JavaScript (Web Workers) e possibilità di utilizzare alcuni siti offline;
-sostituzione del doctype con un semplice <!DOCTYPE html>;

Riferimenti: "Il progetto deve essere basato sulla tecnologia WebRTC, parte delle proposte di evoluzione dell'HTML5".

Link utili:
guide sullo standard HTML5, alcuni esempi:
http://www.guidahtml5.com
http://www.html.it/guide/guida-html5/ (esaustiva e fortemente consigliata)

2.1.6 CSS3

Descrizione: CSS3 è l'ultimo standard per i CSS (Cascading Style Sheets, fogli di stile a cascata), la tecnogia web che associata al linguaggio di (X)HTML permette la definizione delle pagine web. Come è noto, la ragione alla base dell'introduzione dei CSS è rendere quanto più netta la separazione fra i contenuti e le specifiche di formattazione, lasciando al markup (X)HTML la descrizione puramente semantico-strutturale dei contenuti e delegando le istruzioni di formattazione grafica al foglio di stile.

Immediata conseguenza di tale separazione è la maggiore manutenibilità del sorgente (X)HTML, perché le istruzioni di formattazione sono accentrate in un solo punto, sono richiamabili in più punti della stessa pagina (si pensi alle classi di elementi HTML) e possono essere condivise anche fra più pagine se si collega ad esse lo stesso CSS. Ne deriva inoltre un certo incremento prestazionale dal momento che le pagine web diventano di minore dimensione, e occorre dunque meno tempo per il loro download, mentre il CSS può risiedere nalla cache del browser ed avere pertanto tempi di accesso rapidi e senza comportare ulteriori richieste di trasmissione di dati dal server.

In pillole (riguardo tutte le versioni di CSS): una pagina web può riferirsi a un foglio di stile specificando un attributo `style` all'interno degli elementi HTML tradizionali (deprecato in quanto concettualmente equivalente a non avere alcun foglio di stile e mescolare la formattazione direttamente con il sorgente HTML), includendo direttamente il codice all'interno

dell'elemento `<head>` in un elemento annidato `<style>` con attributo `type="text/css"` (questo ha però ancora lo svantaggio di dover inserire il CSS in testa a tutte le pagine che ne fanno uso) oppure, infine, inserendo un elemento di tipo `<link>` sempre nello `<head>` della pagina.

La struttura dei CSS prevede la specifica di una serie di regole per la formattazione dei contenuti. Ciascuna regola è costituita da uno o più selettori seguiti e dal corpo della regola compreso fra parentesi graffe. I selettori si possono distinguere in semplici e combinatori: i primi possono essere generici (`*`), di tipo (riferiscono un elemento HTML), di classe o di ID; i secondi invece permettono di riferirsi a elementi che si trovano in una determinata relazione gerarchica rispetto ad altri elementi (es. `div.content p` si riferisce a tutti gli elementi `<p>` discendenti di un `<div>` di classe `content`), di attributo (es. `a[title]`). Le pseudoclassi e gli pseudoelementi sono altri meccanismi per determinare l'applicazione delle regole sulla base di proprietà. Il corpo di ogni regola, invece, prevede la specifica di una serie di attributi dipendente dal tipo di elemento cui si fa riferimento (es. `color`, `font-weight`, ecc...), ognuno seguito dai due punti e dal valore che si vuole associare ad esso e terminato dai due punti.

In particolare nella versione 3 sono state introdotte alcune novità rispetto alle specifiche precedenti, fra le quali si segnalano

Sintassi:

- selettori di attributo basati su inizio e fine del valore dell'attributo (es. `E[attr~=value]`)
- nuove pseudoclassi (es. `:target` per tutti gli elementi che sono target di un riferimento)

Sfondi:

- opacità di sfondo (`opacity`)
- immagini di sfondo multiple
- sfondi con gradienti

Bordi:

- possibilità di usare un'immagine come bordo
- possibilità di creare bordi arrotondati con attributo `border-radio`

Testo:

- possibilità di caricare font dal server (regola `@font-face`)
- word wrap per evitare l'overflow di testo
- ombreggiature sui caratteri

Altre migliorie:

- possibilità di definire transizioni e animazioni
- selezione di CSS basata sul tipo e sulle caratteristiche del mezzo di visualizzazione con le 'media query'
- trasformazioni lineari su grafica 2D e 3D

Riferimenti: non pervenuti nel capitolato. Inteso in relazione ad una considerazione del gruppo, risulta interessante il loro utillo al fine di gestire l'apparato grafico dell'applicativo, in modo sistematico e performante.

Link utili:
w3schools.com/css3/default.asp
html.it/guide/guida-css3/

2.1.7 Protocolli e funzionalità di Google Chrome

Descrizione:

Il supporto di Google Chrome a WebRTC è realizzato attraverso una serie di API accessibili ai programmi JavaScript:

PeerConnection, che ha lo scopo di permettere la trasmissione in tempo reale di flussi audio/video dall'interno di applicazioni web;

MediaStream, che consente di avere accesso alla webcam e al microfono della macchina su cui il browser è in esecuzione e la cui funzione più significativa è con tutta probabilità il metodo (dal nome abbastanza autoesplicativo) `getUserMedia()` invocabile sull'oggetto `navigator`;

DataChannel, per la trasmissione *peer-to-peer* di flussi dati generici.

Chrome integra il protocollo STUN (e la sua estensione TURN) attraverso il framework ICE (Interactive Connectivity Establishment), che permette, ancora attraverso PeerConnection, di stabilire una connessione *peer-to-peer* anche se gli endpoint della connessione si trovano dietro un NAT.

Supporta inoltre al codec video VP8 e i codec audio iSAC (predefinito), iLBC, G.711, e DTMF. A quanto pare la possibilità di registrare gli stream trasmessi e di condividere lo schermo sono fra le funzionalità che è in programma di integrare nel prossimo futuro (e che potremmo pensare di includere fra i nostri requisiti opzionali o desiderabili).

Per avere accesso a `getUserMedia` occorre abilitare dalla pagina di configurazione `chrome://flag` la funzione sperimentale Ingresso Web Audio (penultima in basso). Per il momento possiamo bellamente ignorare il rassicurante messaggio che compare all'inizio della pagina «Non offriamo assolutamente alcuna garanzia su ciò potrebbe accadere se attivi uno di questi esperimenti: *il tuo browser potrebbe persino andare in autocombustione*!»

Riferimenti: «L'estensione dell'HTML5 WebRTC presente nel browser Chrome si propone di rendere semplice la realizzazione di questi programmi e di far sì che le componenti necessarie siano installate praticamente in ogni computer.»

Link utili:
(generale)
www.html5rocks.com/en/tutorials/webrtc/basics/
(per MediaStream)
dvcs.w3.org/hg/audio/raw-file/tip/streams/StreamProcessing.html
(per MediaStream e `getUserMedia`)
dev.w3.org/2011/webrtc/editor/getusermedia.html
(per PeerConnection)
www.webrtc.org/reference/api-description
(per DataChannel)
dev.w3.org/2011/webrtc/editor/webrtc.html#peer-to-peer-data-api

2.2 Dominio applicativo

Nell'affrontare un progetto è essenziale chiedersi (al fine della fattibilità): qualcuno l'ha già fatto prima? Nel caso di MyTalk la risposta è si. Si riporta che il primo progetto di comunicazione audio/video tramite WebRTC è stato creato dalla Dubango Telecom, e prende il nome di sipML5. SipML5 è il primo client SIP che si basa su WebRTC scritto totalmente in JavaScript e completamente Open Source. La notizia, evidenziata dal sito:

http://www.html5today.it/link/sipml5-primo-client-sip-scritto-interamente-html5

dimostra come il progetto sia fattibile, quanto meno, nella realizzazione dei requisiti obbligatori. Si ribadisce che il progetto sipML5 è open source! Gli sviluppatori ne permettono l'accesso in lettura ai membri non-partecipanti. Per accedere ai sorgenti sarà sufficiente seguire le indicazioni riportate alla pagina:

http://code.google.com/p/sipml5/source/checkout

In merito ad altri applicativi software di voip, è possibile trarre spunti per l'implementazione di requisiti opzionali, analizzando programmi come Skype, seguendo il principio "impariamo dai migliori!".

NOTA AGGIUNTIVA: alla pagina:

http://code.google.com/p/sipml5/wiki/Enable_WebRTC_On_Chrome

è riportato un wiki che evidenzia la possibilità di attivare alcuni flag di google chrome, tra cui anche il WebRTC. Riportiamo la frase: "To enable WebRTC, you need to activate both MediaStream and PeerConnection".

3 Valutazione del capitolato

3.1 Valutazione dei rischi

- Le tecnologie richieste per la creazione dell'applicativo, sono attualmente in via di definizione. La tecnologia WebRTC non è ancora uno standard, e il fatto che le sue API vengano modificate in corso d'opera del progetto, è una certezza. Si pensi che nella data attuale, l'ultima modifica apportata all'architettura risale al 15 novembre. Tuttavia sembrerebbe sensato supporre che le modifiche non intaccheranno in termini distruttivi le basi già istanziate. Questa è ovviamente una congettura personale, che non rappresenta necessariamente la realtà dei fatti. Tuttavia come si evidenzierà nel paragrafo successivo, ciò può comportare anche un punto a nostro favore.

3.2 Vantaggi vari ed evventuali

- Estratto dal capitolato: "In corso d'opera non sarà possibile variare/modificare i requisiti minimi (obbligatori per accettare il prodotto). Sarà invece possibile variare i requisiti opzionali, in quanto saranno i gruppi vincitori dell'appalto a modificarli/eliminarli/aggiungerli". Di conseguenza, non occorre farsi intimorire dall'applicazione del WebRTC, in quanto se si evidenzia l'impossibilità di soddisfare alcuni requisiti obbligatori, potremmo tranquillamente segnalarlo e abbandonarne lo sviluppo. Va da se, che è una pratica sconsigliata. Se mai è più ragionevole partire con meno requisiti opzionali, e aggiungerne in seguito se si crede che il tutto sia fattibile.

- Il software da sviluppare si poggia completamente sul browser Chrome. Di conseguenza si sarà completamente svincolati dalla gestione delle dipendenze dei sistemi operativi sottostanti.

- L'applicativo software poggia le sue funzionalità su un "sito web" che dovrà fungere da server. Per la creazione di tale sito si potrà sfruttare lo spazio web del gruppo, sia per sperimentare il corretto funzionamento delle parti (in fase di sviluppo), sia per permet-

tere al committente di accedere egli stesso all'ambiente, al fine di verificare lo stato del prodotto.

- Riprendendo quanto citato nel paragrafo Valutazione rischi, la possibilità che alcune strutture logiche varino durante lo sviluppo del progetto, imporrà al nostro team di creare un architettura sensata e ben curata. Dando quindi un occhio di riguardo al riuso e alla scalabilità della struttura, incentiveremo l'utilizzo di pattern appropriati e ricercati. Tali potranno risultare punti a nostro favore nel momento della valutazione da parte del docente.
- Come riportato alla voce Dominio Applicativo si evidenzia che il progetto è fattibile. Inoltre è possibile accedere a dei sorgenti che mostrano come operare con la tecnologia WebRTC mediante Javascript.