Università degli Studi di Perugia
Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

Tesi di Laurea

Gestione del frontend per un sistema di visualizzazione di dibattiti

Laureando:
Fabio Paccosi

Relatori:
Prof. Stefano Bistarelli
Prof. Francesco Santini

Anno Accademico 2016–2017

A chi ha sempre creduto in me...

Sommario

Scopo del mio lavoro di tesi è quello di fornire un'interfaccia di gestione del frontend, per gli operatori di una sala di registrazione, ad un sistema di visualizzazione di dibattiti televisivi, che consentirà la creazione di una trasmissione e la gestione della sua visualizzazione e delle relative statistiche sull'andamento e sul testo degli interventi, ricavate utilizzando servizi di Google e IBM. Infine il progetto prevede un'interfaccia per lo spettatore dinamica per consentire la visualizzazione dei risultati di tali elaborazioni mediante grafici in fase di riproduzione in tempo reale o in differita.

Come si vedrà, è stata necessaria la progettazione e lo sviluppo di un'architettura comprensiva di diversi moduli software cooperanti tra loro e la gestione di interfacce dinamiche in base al contesto trattato.

Inoltre, sarà evidenziato lo studio effettuato per rendere coerente la visualizzazione dei dati ottenuti dall'elaborazione del testo, che è stato basato su molti casi studio reali e sulle tecnologie utilizzate per l'implementazione di tali grafici.

Indice

Capitolo 1

Introduzione

Considerato il crescente numero di dibattiti presenti ogni giorno nelle reti televisive e l'attuale staticità nella visualizzazione delle stesse, si è pensato di fornire un servizio aggiuntivo che consentisse una maggiore qualità e quantità di informazione. Pertanto in questa tesi è stato proposto un sistema di gestione frontend[1] in grado di unire il mondo dell'informatica e quello televisivo al fine di mostrare all'utente finale informazioni specifiche sull'andamento di un dibattito, come i sentimenti provati dagli interlocutori, i toni di ogni singolo intervento e tante altre caratteristiche.

Il progetto di tesi è stato diviso in due parti. Nella prima parte la gestione del backend[2] ha previsto la creazione di un sistema in grado estrapolare, tramite apposito hardware, le tracce audio da una sala di registrazione e di analizzare il contenuto dei file ricavandone informazioni visualizzabili attraverso un'analisi del testo fornita da servizi esterni. Nella seconda parte,

[1]La parte visibile all'utente e con la quale si può interagire.

[2]La parte nascosta che permette l'effettivo funzionamento di queste elaborazioni.

il cui contenuto viene descritto in questa tesi, è stata progettata la parte frontend del il sistema che ha previsto gli aspetti della configurazione e della gestione di un dibattito ed, infine, l'implementazione del software per la visualizzazione dei dati analizzati dal software di backend attraverso un aggiornamento dei dati in tempo reale.

In questo documento verranno descritti tutti gli aspetti coinvolti nello sviluppo del sistema di gestione fronted e di quello di visualizzazione di dibattiti.

Il capitolo 2 conterrà la descrizione generale del progetto, con un'analisi del contesto applicativo e degli obiettivi da raggiungere. Inoltre verranno elencati i software che saranno implementati e tutte le loro caratteristiche, i modelli di sviluppo utilizzati per la progettazione e la specifica dei requisiti e delle funzionalità richieste e le rispettive fasi di testing e debugging.

Nel capitolo 3 verrà esposto lo studio effettuato sulle varie tipologie di dibattiti esistenti e come l'intero sistema sia stato modellato su questa ricerca. Verranno inoltre riportati dei casi di studio e degli studi accademici correlati che hanno permesso la progettazione di grafici di facile lettura per l'interfaccia visiva dell'utente finale.

Il capitolo 4 conterrà la spiegazione tecnica dettagliata dell'intera infrastruttura del software: verranno elencati i vari moduli operativi, le loro funzionalità e come sono stati sviluppati. Inoltre saranno presenti le descrizioni di particolari librerie e delle tecnologie web utilizzate, una panoramica sulla struttura del database e come le informazioni verranno mostrate dalla GUI[1] del sistema di visualizzazione. Inoltre verrà fornita una panorami-

[1]L'interfaccia grafica utente (dall'inglese Graphical User Interface).

ca degli strumenti utilizzati in fase di progettazione grafica e sviluppo del software, mostrandone le caratteristiche fondamentali che hanno aiutato maggiormente in considerazione alle tecnologie utilizzate per la creazione del sistema.

Nell'ultimo capitolo saranno analizzati progetti esistenti correlati al nostro sistema di gestione di dibattiti e verranno elencati una serie di possibili sviluppi futuri.

Capitolo 2

Descrizione del progetto

2.1 Contesto

I software realizzati per questo progetto sono sviluppati per una qualsiasi sala di registrazione televisiva, ossia per coloro che dispongono dei giusti mezzi per la trasmissione di un dibattito come una scheda audio[1] capace di gestire indipendentemente i canali[2] recuperati e uno o più elaboratori nei quali sia installato un browser vista la natura web oriented dell'intera architettura. Inoltre, la gestione e configurazione del sistema è stata adattata per le diverse tipologie di trasmissione che qualsisi rete offre e, in fase di visualizzazione delle statistiche, i grafici sono stati progettati per rendere fruibili e facilmente consultabili le informazioni da parte dell'utente.

[1] E' una scheda di espansione per computer che si occupa di elaborare un flusso audio digitale in input in un segnale analogico o digitale da inviare in output ad una periferica audio o viceversa.

[2] Flusso informativo elettronico contenente informazioni prese in input.

2.2 Scopi e obiettivi del progetto

Scopo del progetto è quello di fornire un'infrastruttura software completa che consenta la creazione, la gestione e la configurazione di un dibattito in tutti i suoi aspetti. Inseriti i dati nel sistema ed estrapolati quelli da analizzare, questo dovrà interfacciarsi con dei servizi esterni di Google Cloud Speech[1] che permette agli sviluppatori di convertire un file audio in testo e IBM Watson[2] che fornisce un grande numero di strumenti per l'analisi del testo. Una volta analizzati i dati, il sistema sarà quindi in grado di recuperare i risultati e visualizzare sullo schermo degli utenti le statistiche indipendentemente dalla modalità di trasmissione del dibattito (real time[3] o differita[4]).

2.3 Software richiesti

Per rendere l'intera infrastruttura il più intuitiva possibile e di facile utilizzo, si è scelto di realizzare diversi moduli software[5], ognuno dedicato ad un compito specifico per la configurazione e la visualizzazione di una trasmissione: un client web[6] sarà utilizzato per creare l'evento, inserire i suoi partecipanti e definire alcune informazioni di base come le credenziali dei servizi esterni autilizzati mentre, sempre utilizzando tecnologie web, un al-

[1] `https://cloud.google.com/speech/`.

[2] `https://www.ibm.com/watson/`.

[3] Processo che prevede l'immediata trasmissione ed elaborazione dei dati.

[4] In questo tipo di trasmissione dati le informazioni vengono richieste dopo il recupero.

[5] E' un componente software autonoma ben identificata.

[6] Componente accedibile tramite browser web per richiedere risorse.

tro client sarà dedicato alla configurazione visuale dei grafici mostrati in fase di trasmissione e dei colori utilizzati dai vari partecipanti. Successivamente, un altro software potrà connettersi al dibattito desiderato ed iniziare così la sua visualizzazione e visualizzare i grafici con le statistiche frutto dell'analisi del testo effettuata in backend.

Ricapitolando, avremo tre diversi client per la sola parte di frontend, di cui saranno dedicati ad operatori della sala di registrazione mentre lo spettatore utilizzerà solo il client per la visualizzazione come detto poc'anzi.

C'è poi tutta la parte backend, dove avviene l'effettiva analisi dei dati ed il loro salvataggio nel database del sistema: in questa fase un server, tramite software scritto in Python, estrapola dai diversi canali audio il testo dell'intervento sfruttando degli appositi servizi di Google, analizza il testo, questa volta utilizzando le librerie IBM Watson, e salva i risultati all'interno del database, comunicando al client di visualizzazione un aggiornamento nei dati a disposizione per il relativo dibattito. Tutte le informazioni, comprese quelle di configurazione dei grafici, vengono salvate su database e qui resteranno a disposizione per visualizzazioni in differita future. Si vede dalla figura 2.1 il ciclo di operatività dei diversi moduli software e come questi intervengo durante la configurazione e la visualizzazione di una trasmissione.

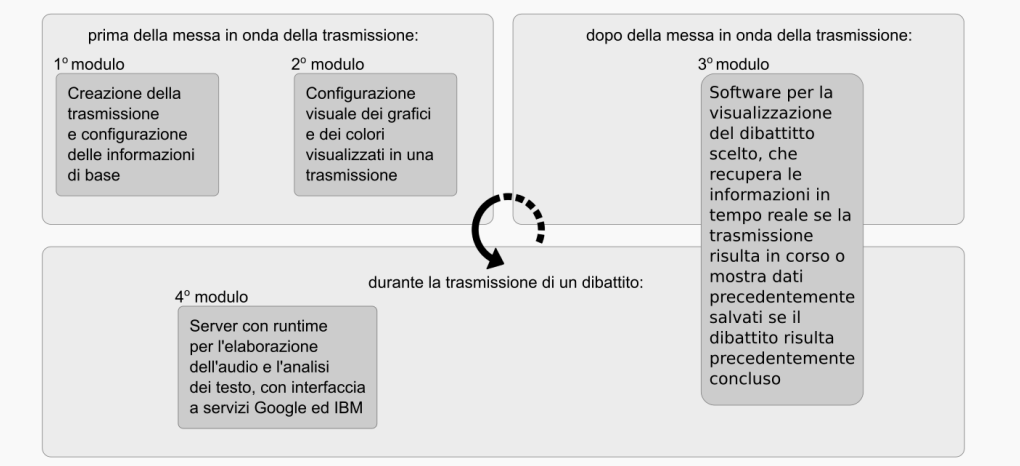

Figura 2.1: L'immagine rappresenta i vari moduli software nel tempo in cui sono operativi.

2.4 Analisi dell'audio e del testo estrapolato

Come già detto nei precedenti paragrafi, l'operatività di tutto il sistema, per quanto riguarda il recupero di dati statistici da visualizzare, si basa sull'utilizzo di due servizi esterni, uno sviluppato da Google e l'altro da IBM.

Google Cloud Speech permette agli sviluppatori di convertire una sorgente audio in testo applicando, inoltre, modelli di rete neurali per l'apprendimento al fine di aumentare nel tempo la precisione e l'affidabilità del riconoscimento del testo. Questo servizio riconosce oltre 110 linguaggi diversi e consente uno stream dei risultati in tempo reale.

Il servizio di IBM si chiama Watson, nome dato in onore del primo presidente dell'IBM Thomas J. Watson, ed è un sistema di intelligenza artificiale

in grado di rispondere a quesiti espressi in un linguaggio naturale[1] sviluppato all'interno del progetto DeepQA[2]. Watson è un'applicazione avanzata di elaborazioni del linguaggio e rappresentazione della conoscenza anche questa basata su tecnologie di machine learning, cioè di apprendimento automatico.

2.5 Specifica dei requisiti e modello di sviluppo

Per definire le caratteristiche dei moduli del software è stato necessario uno studio della stuttura dei vari tipi di dibattiti, così da implementare tutte le funzionalità necessarie e visualizzare dati coerenti in fase di trasmissione. Sono state selezionate, come prima cosa, le varie tipologie di dibattiti come interviste e talk televisivi, decisi quali aspetti degli interventi caratterizzare e confrontare (emozioni, toni, sentimenti etc.) e come poterli visualizzare in maniera coerente e senza perdita di generalità tra le diverse tipologie di trasmissione e, in seguito, sono state selezionate le informazioni desiderate tra la molteplicità di dati restituiti dai servizi esterni di analisi del testo di IBM. Successivamente si è passati alla progettazione dell'architettura software[3] dell'intero sistema e, tramite uno studio rigoroso delle funzionalità da implementare, si è scelto come suddividere i compiti tra le varie componenti del progetto. Per quanto riguarda l'aspetto ingegneristico dello sviluppo si è

[1] Lingua parlata da una comunità linguistica e formatasi con l'evoluzione storica.

[2] `https://www.research.ibm.com/deepqa/deepqa.shtml`.

[3] L'organizzazione fondamentale di un sistema, definita dai suoi componenti, dalle relazioni reciproche tra i componenti e con l'ambiente.

scelto un modello evolutivo[1] basato sul modello waterfall[2] al fine di ottenere prototipi throw-away[3] per capire meglio quali requisiti sono stati poco specificati e tutte le funzionalità indispensabili da implementare. Ogni iterazione del ciclo dello sviluppo è composto, come si vede dalla figura 2.2, da fasi di definizione dei requisiti, design dell'implementazione tramite il linguaggio di modellizzazione UML[4], sviluppo e test, l'eventualmente l'integrazione nel sistema software delle nuove funzionalità ed infine il mantenimento.

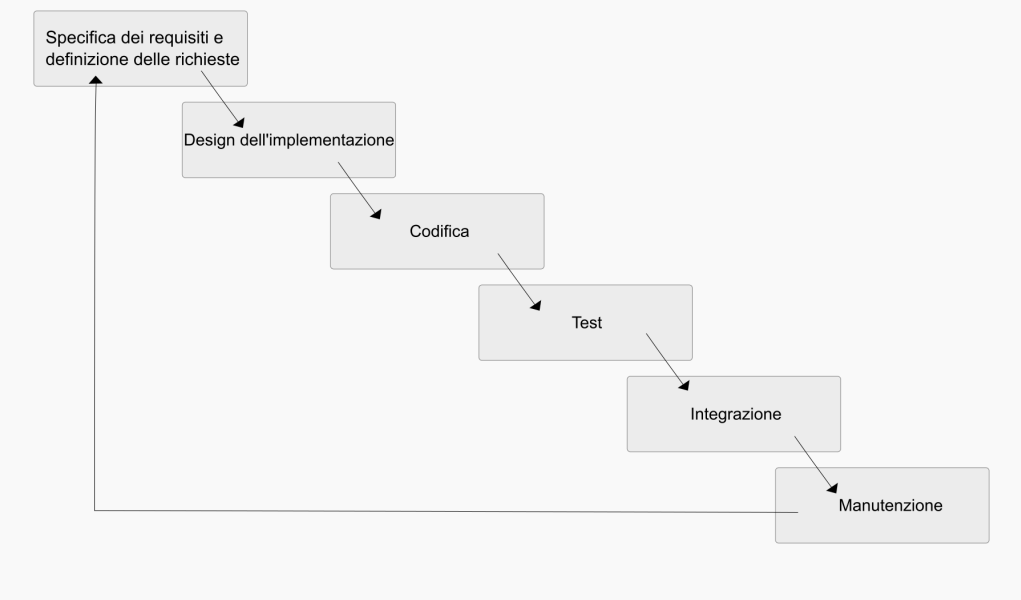

Figura 2.2: Esempio di un'iterazione del modello di sviluppo incrementale.

Si è scelto di utilizzare questo modello di sviluppo in maniera iterativa poichè non tutte le specifiche da implementare nel sistema erano note allo

[1]Modello del ciclo di vita del software che fornisce strumenti per superare i limiti principali del modello a cascata.

[2]È il più tradizionale modello del ciclo di vita di un software.

[3]Un prototipo "cestinabile" che fornisce dei feedback per il prototipo successivo.

[4]`http://www.uml.org`.

stesso tempo e sono spesso intervenute variazioni dei requisiti durante il lavoro. Per questo motivo, una metodologia di progettazione e realizzazione di tipo incrementale ha permesso la possibilità di gestire queste criticità proprie del tradizionale modello "a cascata" dato che l'intero sviluppo è stato basato sull'evoluzione e l'aggiornamento progressivo di prototipi intermedi e non dell'architettura complessiva del sistema.

2.6 Testing e debugging

Una volta effettuata la verifica[1] del software si è passati alla fase di testing[2] e di debugging[3] degli applicativi del sistema. A tal proposito, prima di provare il software con trasmissioni in tempo reale, si è scelto di utilizzare dati prelevati da alcuni dibattiti precedentemente svolti (ad esempio delle sbobinature[4] di interviste e di Consigli Comunali). Inserite tutte le informazioni riguardanti la tipologia della trasmissione e quelle relative ai partecipanti, si è creata l'interfaccia televisiva di prova e sono state chiamate le librerie esterne per l'analisi del testo trascritto, in tal modo è stata corretta la maggior parte degli errori in fase di visualizzazione delle statistiche. Successivamente sono state svolte le prove in tempo reale per verificare la correttezza dell'elaborazione lato server e sono stati risolti gli errori rileva-

[1]Insieme delle attività volte a stabilire se il programma costruito soddisfa le specifiche (non solo funzionali).

[2]Verifica fatta mediante esecuzione del programma, selezionando a priori dati di ingresso e valutando la correttezza risultati.

[3]Localizzazione ed eliminazione degli errori nel codice.

[4]Trascrizione del contenuto di una registrazione.

ti. Per questa tipologia di prova è stato utilizzato hardware esterno, per la precisione una scheda audio professionale che ha garantito la possibilità di lavorare su canali audio e con microfoni differenti in modo tale da simulare, il più verosimilmente possibile, una sala di registrazione televisiva.

Capitolo 3

Studio delle tipologie di dibattiti e dello stato dell'arte nei software di visualizzazione

3.1 Lo stato dell'arte

Diverse sono le applicazioni di sistemi per la visualizzazione di informazioni ricavate dall'analisi di dibattiti e lo scopo per il quale vengono utilizzate. Il maggior contributo in materia è stato dato dalle ricerche di ARG-tech[1] che, applicando gli studi teorici sull'argomentazione di dibattiti, ha sviluppato diversi progetti, tra i quali l'interessante Argument Analysis Wall[2]: questo progetto ha come obiettivo quello di analizzare dibattiti e supportare la consultazione degli argomenti trattati online grazie ad analisti che,

[1] `http://www.arg-tech.org/`.

[2] http://www.arg-tech.org/index.php/projects/argument-analysis-wall.

come si vede nella figura 3.1, sfruttano un apposito hardware, chiamato AnalysisWall, per creare collegamenti in tempo reale tra gli argomenti del dibattito.

Figura 3.1: Il lavoro degli analisti di ARG-tech nella creazione e nel collegamento di argomenti in real time.

Un altro portale che si prefigge lo scopo di creare una banca dati di dibattiti e di argomenti correlati è Debate Hub[1] sviluppato da Knowledge Media Institute[2]: si tratta di una community online molto attiva che utilizza tecnologie avanzate per la catalogazione, l'analisi e la visualizzazione di argomenti come si può vedere dalla figura 3.2.

[1] `https://debatehub.net`.

[2] `http://idea.kmi.open.ac.uk`.

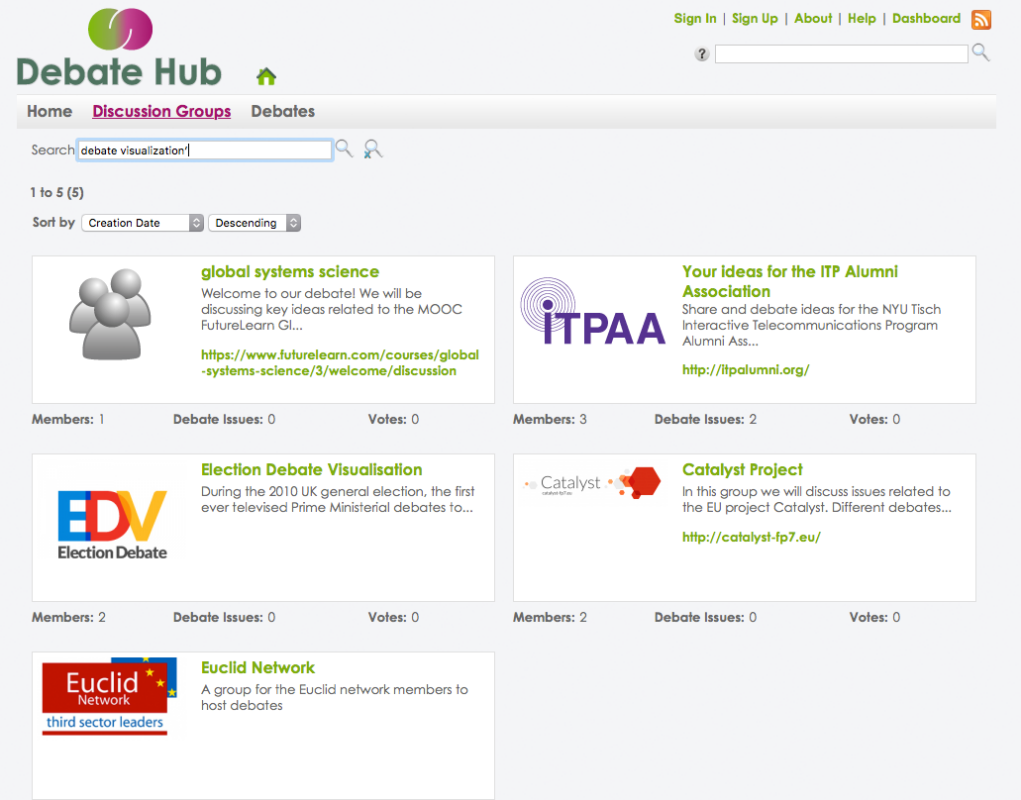

Figura 3.2: Esempio di catalogamento e ricerca per argomenti di Debate Hub.

DebateGraph[1] è invece una piattaforma online che aiuta la sua community a esternalizzare, visualizzare, interrogare e valutare tutte le considerazioni di un membro del dibattito in questione che possano essere rilevanti per l'argomento, ad esempio generando mappe logiche come si vede in figura 3.3, e facilitando un dialogo all'interno della community stessa attorno a tale tema.

[1] `http://debategraph.org/`.

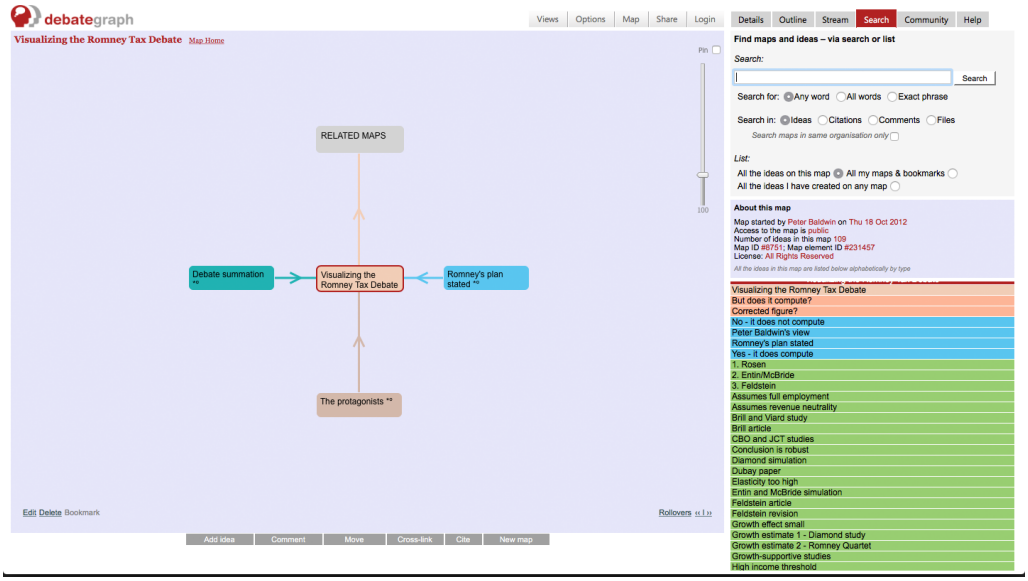

Figura 3.3: Mappa degli argomenti di un dibattito genera da DebateGraph.

Infine il progetto EDV[1], acronimo per Election Debate Visualisation, è stato finanziato dal Engineering & Physical Sciences Research Council[2] (EPSRC) del Regno Unito. Sviluppato in 3 anni e lanciato nell'ottobre 2013, il programma è stato condotto dall'Università di Leeds[3] (Scuola di Media e Comunicazione) e dall'Università Open[4] (Knowledge Media Institute). EDV

[1] `http://edv-project.net`.

[2] `https://www.epsrc.ac.uk`.

[3] `http://media.leeds.ac.uk`.

[4] `http://kmi.open.ac.uk`.

Figure A1 Party voting intentions over the campaign
(Source: YouGov polls)

Figura 3.4: Esempio dei risultati ottenuti da EDV nell'analisi della campagna elettorale ministeriale britannica del 2010.

Come si evince dalla figura 3.4, EDV ha analizzato come i cittadini britannici hanno elaborato i dibattiti televisivi dei candidati alla carica di Primo Ministro nel 2010, al fine di comprendere come i dibattiti del 2015 possano essere sperimentati in nuovi modi per dare maggiore comprensione e appetibilità. Questi risultati sono stati utilizzati per sperimentare l'uso di nuovi approcci computazionali per rappresentare ciò che avviene nei dibattiti, attraverso nuove interfacce visive di replay.

3.2 Lo studio delle tipologie di dibattito

Accedendo ad una qualsiasi rete televisiva ci si rende subito conto di come vengano trasmessi una grande varietà di dibattiti[1] che di come questi ven-

[1] `http://www.garzantilinguistica.it/ricerca/?q=dibattito`.

gano differenziati da una serie di caratteristiche che ne valorizzano alcuni aspetti piuttosto che altri, come ad esempio le categorie di argomentazione[1] che possono essere le più disparate così come il numero di partecipanti, il loro ruolo all'interno della trasmissione e il loro raggruppamento .

In fase di studio per la creazione di questo sistema di visualizzazione delle statistiche sull'andamento di un dibattito ho deciso di raggruppare tutte queste caratteristiche in diverse macrosezioni e, studiando diverse trasmissioni realmente svolte nel tempo, ho trovato alcuni punti in comune tra quelle simili e distinto le altre.

Uno tra i casi di studio più importanti è stato, nello specifico, la recente elezione presidenziale americana con tutti dibattiti e le interviste ad essa collegate: in particolare gli studi statistici effettuati da ARG-tech e realizzati per conto dell'Università di Dundee, dei quali potete vedere un esempio nella figura 3.5 che mostra il tempo di inizio degli interventi, la percentuale di partecipazione di ogni interlocutore e dei confronti sull'andamento, sono stati di grande aiuto per capire come poter elaborare e visualizzare delle informazioni estrapolate da una specifica tipologia di trasmissione.

[1] `http://www.storyboardthat.com/it/articles/e/discussione`.

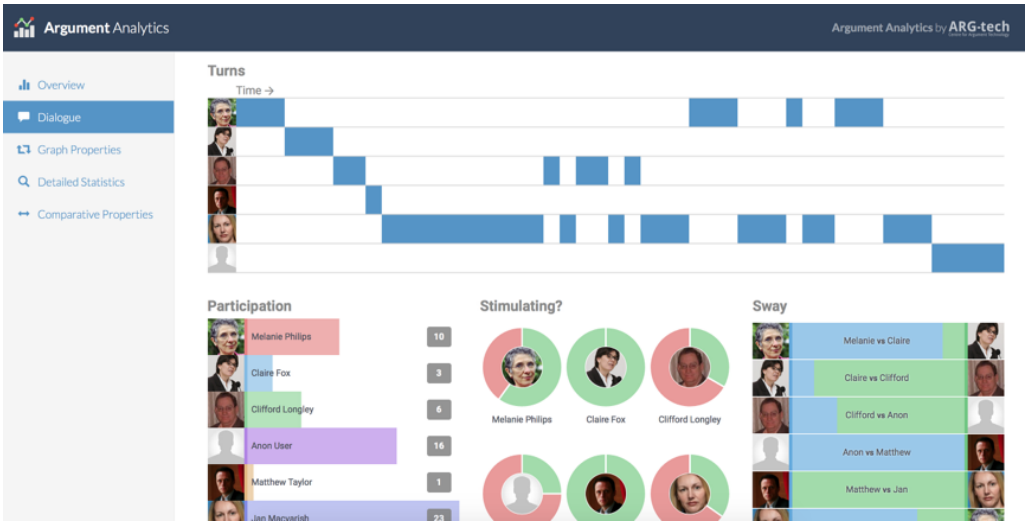

Figura 3.5: Un esempio di Argument Analytics di ARG-tech dal sito ufficiale del progetto.

Prendendo spunto proprio dalle elezioni presidenziali americane, la prima tipologia utilizzata per differenziare la grande varietà di trasmissioni è stata quella del 1 Vs. 1 (uno contro uno o duello televisivo/radiofonico). In questo tipo di dibattito due sfidanti si confrontano in un duello che può essere o meno mediato da un conduttore, in una variante più diretta del classico dibattito (o tribuna), ove più di due partecipanti interagiscono sotto la conduzione di un elemento neutro, ciò diversamente da quanto avviene, ad esempio in un talk (o salotto) dove i partecipanti si alternano senza la presenza di un conduttore. Esiste inoltre una tipologia di dibattiti che prevede la presenza di partecipanti raggruppati in team che si confrontano sui vari argomenti similmente a come avviene in un classico talk. Alla famiglia dei dibattiti/talk, si affianca quelle delle interviste, che si differenziano l'una dalle altre per tipologia di conduzione e svolgimento: ad esempio, in un'intervista strutturata si conosceranno a priori le domande che verranno

rivolte all'intervistato, mentre in un'intervista semi strutturata il conduttore alternerà domande/argomenti prefissati ad altri da lui improvvisati; nella tipologia non strutturata, invece, non c'è niente di deciso a priori e l'intero svolgimento viene improvvisato dall'intervistatore.

Ricapitolando, le tipologie selezionate da questa ricerca su diversi casi di studio reali sono:

- 1 Vs. 1
- Dibattito
- Talk
- Team Vs. Team
- Intervista strutturata
- Intervista semi strutturata
- Intervista non strutturata

Le differenze tra le singole trasmissioni si ripercuotono indubbiamente sulla tipologia di dati statistici estrapolati e sulla loro visualizzazione: infatti alcuni tipi di grafico avranno senso soltanto se visualizzati in presenza di determinati dibattiti, oppure la visualizzazione degli stessi dati potrebbe essere ugualmente possibile, anche se in forma diversa. Ad esempio, in un dibattito Team Vs. Team, avrà senso inserire un grafico che ci illustra l'andamento dei diversi team con un corrispettivo grafico, magari di diverso aspetto, per le altre tipologie di trasmissione che non contemplano la presenza di team. Similmente, alcuni grafici potranno essere in comune ed avere

la stessa forma per più tipologie di dibattiti, come i grafici nella figura 3.6, che rappresentano la timeline degli interventi, l'andamento dei partecipanti e delle domande nel tempo trascorso, cioè tutte informazioni facilmente interpretabili per la maggior parte delle situazioni.

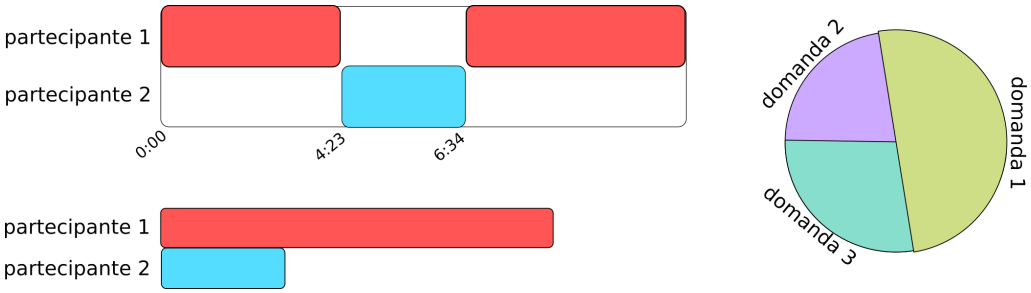

Figura 3.6: Esempio di grafici: in alto a sinistra una timeline degli interventi, in basso a sinistra l'andamento dei partecipanti e a destra l'andamento delle domande di un'intervista.

Un altro aspetto sul quale ci siamo soffermato per identificare meglio il tipo di grafico e le informazione mostrate è quello dei colori: in fase di configurazione sarà possibile scegliere un colore identificativo per ogni partecipante (o team) e questo verrà utilizzato per mostrare ed identificare con più facilità le informazioni correlate allo specifico partecipante. In figura 3.7 si può notare uno studio preliminare (mockup, cioè un'immagine statica di come apparirà il prodotto finale) della visualizzazione di un dibattito televisivo. Questo tipo di render vettoriali ha aiutato molto nella progettazione e nell'ottimizzazione dello spazio dei diversi grafici in ogni fase dello sviluppo.

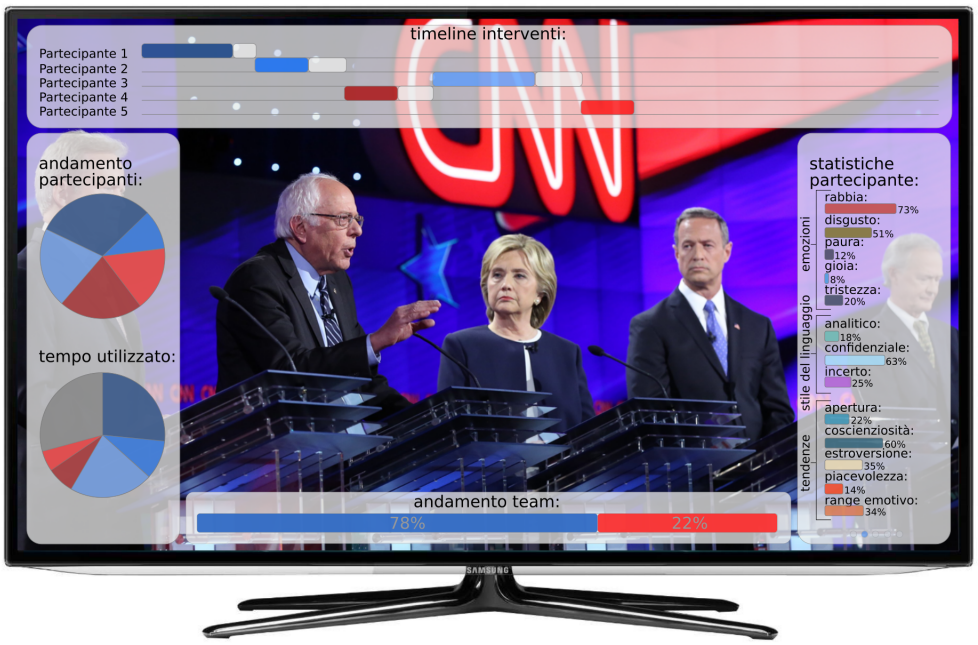

Figura 3.7: Mokup iniziale basato su una tipologia di dibattito Team Vs. Team.

Capitolo 4

Infrastruttura software

L'infrastruttura software, così come si può vedere nella figura 4.1, prevede, nella sola fase del frontend per la gestione e la messa in onda di un dibattito, due pannelli di controllo differenti: il primo consente una configurazione iniziale delle informazioni principali (come il nome dato al dibattito, la data e la tipologia di trasmissione e i dati relativi ai partecipanti) mentre il secondo permette di mettere a punto e personalizzare la visualizzazione delle statistiche e tutti gli aspetti grafici dell'interfaccia in sovraimpressione mostrata durante la trasmissione. Mentre un dibattito è in onda, le runtime del server backend scritte in Python, che elaboreranno le registrazioni e il testo, inviano i dati alle APIs dei servizi di IBM Watson e Google Speech che, in tempo reale, provvedono ad estrapolare il testo da una sorgente audio ed analizzarlo utilizzando i più evoluti progressi scientifici in materia di text analysis[1], così che il client di visualizzazione possa essere in grado di disegnare i grafici e mostrare le informazioni ottenute durante l'elaborazione.

[1] `https://www.ibm.com/watson/developercloud/doc/personality-insights/science.html`.

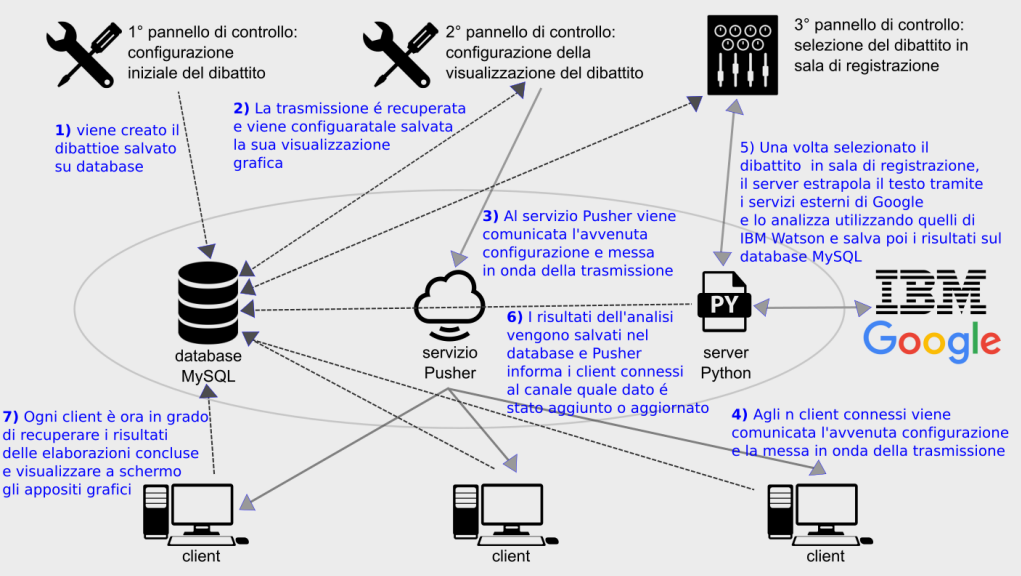

Figura 4.1: Schema dell'infrastruttura del software.

4.1 I pannelli di controllo

I due pannelli di controllo per la gestionde del frontend consentono di gestire diversi aspetti della configurazione di una trasmissione: il primo permette l'inserimento delle informazioni di catalogazione di un dibattito (nome, data, tempo stimato etc.), i dati anagrafici dei rispettivi partecipanti e le credenziali di accesso ai servizi esterni che il sistema utilizza, mentre il secondo è dedicato alla configurazione visuale dei grafici per le statistiche elaborate in cui gli operatori possono scegliere sia quali risorse mostrare a video sia dove posizionarle, oltre ad impostare altri parametri specifici per ogni singolo grafico.

4.1.1 Configurazione del dibattito

Il primo pannello di configurazione del frontend ci permette di creare un nuovo dibattito, scegliendone la tipologia, la lingua che si parlerà ed impostando le informazioni anagrafiche sui partecipanti, l'eventuale presenza di team (in base alla scelta effettuata sulla tipologia della trasmissione), l'esistenza o meno di parole chiave da ricercare durante la trasmissione in fase di analisi degli interventi e le credenziali per l'accesso ai servizi esterni di analisi del testo e di traduzione utilizzati dal sistema.

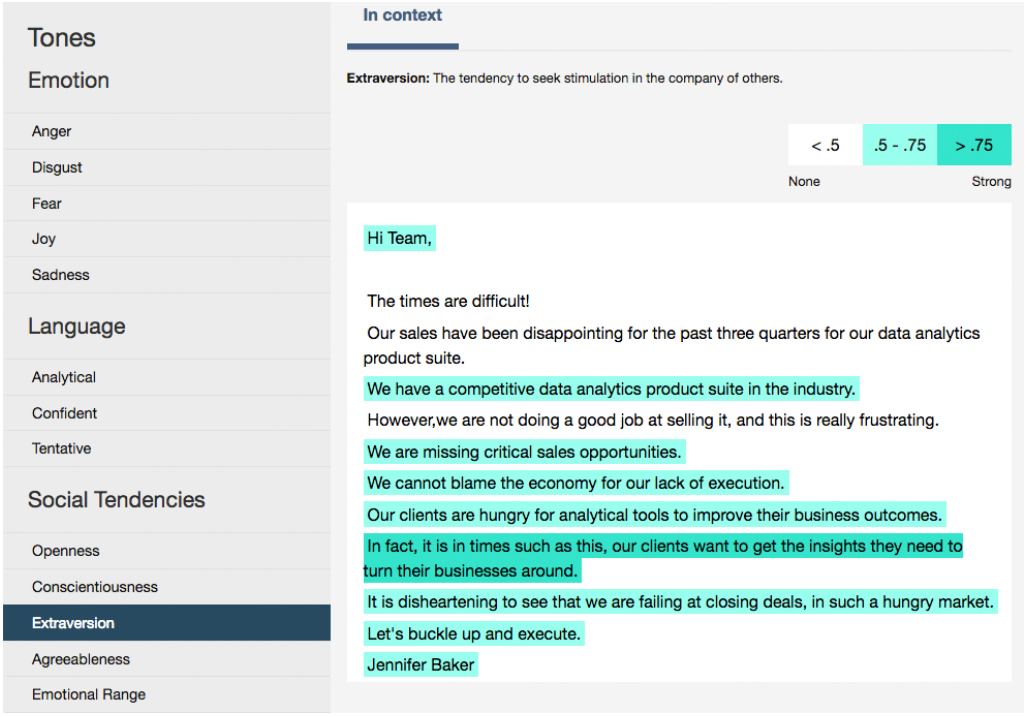

Figura 4.2: Esempio dell'analisi di un testo da parte del servizio di IBM chiamato Tone Analyzer.

La possibilità di scegliere la lingua del dibattito consente, se ci troviamo di fronte ad una trasmissione svolta in lingua inglese (o qualsiasi altra lingua

supportata dai servizi), di saltare la fase di traduzione del testo. Questo è infatti un passaggio a volte necessario poichè alcune librerie di analisi possono ricevere in input solamente un determinato tipo di idioma, rendendo quindi necessaria la traduzione in tempo reale di un intervento per recuperarne i dati.

Inoltre, questo pannello di controllo permette di visualizzare tutti i dibattiti precedentemente configurati, quelli attualmente in onda e quelli che risultano conclusi. Le informazioni inserite verranno salvate su database, così da poter essere recuperate in seguito per la successiva configurazione del frontend per l'aspetto grafico e la messa in onda della trasmissione.

4.1.2 Configurazione visuale dell'interfaccia grafica

Una volta creato il dibattito si passa, tramite il secondo pannello di controllo del frontend, alla configurazione dei grafici che mostreranno le statistiche sul suo andamento

Da un menù a tendina si potrà scegliere il dibattito, tra quelli precedentemente inseriti nel database, cosicchè, in base alla scelta effettuata, il sistema consentirà o meno la personalizzazione di determinati tipi di informazioni (ad esempio, in un dibattito di tipo team Vs. team, potremo scegliere i colori che rappresentano ciascuna squadra di partecipanti).

Si deciderà, inoltre, se la visualizzazione della trasmissione sarà protetta o meno da una password di accesso. A questo punto potremo scegliere con quale colore verranno identificati i vari partecipanti, le eventuali parole chiave ed i team, gestire la presenza di determinati grafici. Inoltre, come si può vedere dalla figura 4.3, potremo gestire la posizione di tutti gli elementi

tramite un'anteprima di visualizzazione con funzionalità drag and drop[1], e creare così l'interfaccia grafica desiderata e condivisa da tutti i client che visualizzeranno, in diretta o in differita, la trasmissione.

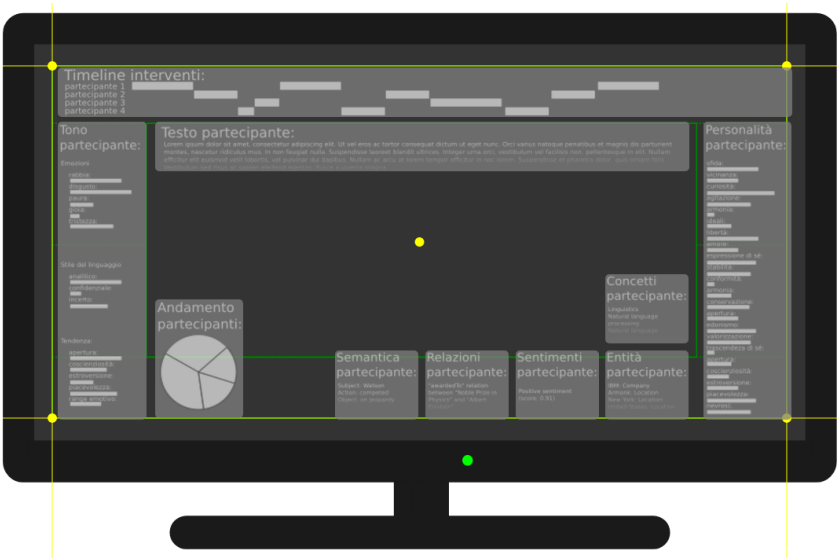

Figura 4.3: Esempio dell'anteprima della visualizzazione di una trasmissione in fase di configurazione.

4.2 Il client di visualizzazione

Una volta collegati al client di visualizzazione, si aprirà una schermata in cui sarà possibile scegliere il dibattito che si desidera visualizzare: se quest'ultimo è protetto, verrà richiesta una password, altrimenti si procederà immediatamente alla sua visualizzazione. Inoltre da questa schermata iniziale è possibile visualizzare alcune informazioni preliminari della trasmissione

[1]Quando si parla di interfacce grafiche, si intende una successione di tre azioni, consistenti nel cliccare su un oggetto virtuale (quale una finestra o un'icona) per trascinarlo (in inglese: drag) in un'altra posizione, dove viene rilasciato (in inglese: drop).

che si vuole visualizzare, come la data di svolgimento o, ad esempio, il suo stato (iniziata o terminata).

Ad accesso avvenuto, il sistema controllerà se la visualizzazione dei grafici è pronta per essere scaricata; in caso positivo è possibile attingere alle informazioni presenti nel database (posizione dei singoli elementi ed altre configurazioni) per istanziare i vari grafici. Recuperate le informazioni, viene elaborata l'interfaccia per l'utente in base alle caratteristiche date in fase di configurazione, alle proprietà della specifica trasmissione e dei suoi partecipanti. Il sistema si metterà ora in ascolto di messaggi di richiesta di aggiornamento delle informazioni che dovranno essere visualizzate.

4.2.1 Dibattiti real time o in differita

In base allo stato della trasmissione (in onda, terminata o non ancora iniziata) il sistema si comporta diversamente: se la trasmissione selezionata dovesse risultare ancora non iniziata, il client si mette in ascolto di un messaggio di avvenuta messa in onda, così da ottenere poi tutti gli aggiornamenti (in tempo reale) dalle altre componenti dell'infrastruttura dedicate alla manipolazione dell'audio, all'analisi del testo e salvataggio su database.

Se il dibattito risultasse invece terminato oppure iniziato precedentemente e non ancora concluso, il sistema scarica tutte le informazioni frutto delle precedenti elaborazioni, le visualizza e resta in ascolto di ulteriori aggiornamenti e notifiche sulla presenza di ulteriori dati.

Le richieste che il client farà al database sono fondamentalmente le stesse, cambieranno semplicemente i messaggi dei quali il client si metterà in ascolto, la sincronia tra i vari moduli dell'architettura ed il tempo di esecuzione

delle query[1].

4.2.2 Recupero delle statistiche

Tutti i dati visualizzati e le configurazioni vengono salvate su database. In fase di salvataggio, il server comunica all'applicazione che ha spedito le informazioni il responso positivo dell'operazione, così da poter iniziare il dialogo, se necessario, con le altre componenti del sistema.

Per recuperare i dati in tempo di esecuzione da parte del client, vengono utilizzate delle classiche query PHP, come si può vedere nella figura 4.4.

```php
//Recupera i grafici di un dibattito
function get_charts ($debate_id){
    $servername = "localhost";
    $username = "debatevis";
    $password = "VediTu";

    try {
        $db = new PDO("mysql:host=$servername;dbname=debatevis;charset=utf8", $username, $password);
        $query = "select * from chart where debate_id = $debate_id";

        //query result
        $charts = array();
        $sth = $db->query($query);
        while( $row = $sth->fetch(PDO::FETCH_ASSOC) ) {
            $charts[] = $row; // appends each row to the array
        }
        return $charts;
    }
    catch(PDOException $e){
        echo "Connection failed: " . $e->getMessage();
    }
}
```

Figura 4.4: Esempio di query in PHP utilizzata per recuperare tutti i grafici di un dibattito selezionato.

Tutta la procedura di ascolto ed invio di messaggi è implementata, fa-

[1]L'interrogazione di un database per estrarre o aggiornare i dati che soddisfano un certo criterio di ricerca.

cendo collaborare lato server[1] e lato client[2] dell'infrastruttura, utilizzando AJAX[3] e Javascript per l'interattività e dinamicità dell'applicazione e PHP per query e manipolazioni di dati.

L'operatività congiunta di questi diversi linguaggi di programmazione è data da funzioni tipiche di richiesta, tramite protocollo L'HyperText Transfer Protocol [3] (HTTP, protocollo di trasferimento di un ipertesto), tipico di un'architettura client-server[4] del genere.

```
function js_update(debate_id, type){
    query_data = "debate_id="+String(debate_id)+"&type="+type;
    //Participations
    $.ajax({
        type: "POST",
        url: "src/client/server_interface/update_db.php",
        data: query_data,
        dataType: "json",
        success: function(msg,string,jqXHR){
            alert(msg);
        }
    });
}
```

Figura 4.5: Esempio di richiesta di aggiornamento di tipo POST al server da parte di una funzione client side.

[1]Si fa riferimento a operazioni compiute dal server in un ambito client-server

[2]Le operazioni di elaborazione effettuate da un client in un'architettura client-server. Rappresenta dunque il frontend di un sistema informatico e di un'applicazione web con architettura multi-tier.

[3]Acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive e multi-piattaforma.

[4]Un'architettura di rete nella quale genericamente un computer client o terminale si connette ad un server per la fruizione di un certo servizio.

Tutte le richieste al server vengono effettuate con il metodo POST[1], il quale è usato di norma per inviare informazioni al server (usualmente in presenza di informazioni sensibili). In questo caso l'URI [4] del messaggio indica che cosa si sta effettivamente inviando e il corpo del messaggio ne esplica il contenuto.

```php
<?php
    $debate_id = $_REQUEST["debate_id"];
    $type = $_REQUEST["type"];


    require_once 'get_participation.php';//Recupera tutti gli interventi
    $participations_new = get_all_participation($debate_id);
    $r = json_encode($participations_new);


    echo $r;
?>
```

Figura 4.6: Esempio dell'elaborazione di una risposta per la richiesta del client da parte del server.

I messaggi scambiati tra client e server sono scritti nella sintassi JSON [5] (JavaScript Object Notation), vista la naturale rapidità nell'interpretazione e nel parsing delle architetture web based per il quale questo formato è nato. Come da prassi, i dati di tipo JSON restituiti da una qualsiasi risposta HTTP hanno, per correttezza, un'intestazione Content-Type. Nel caso di specie, il JSON, che supporta nativamente sia Array che Array associativi[2], è stato davvero utile per facilitare la corretta trasmissione delle numerose

[1] `https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html`.

[2] L'array associativo è un array i cui elementi sono accessibili mediante diversi tipi di indice tra cui nomi, quindi stringhe anziché indici puramente numerici.

informazioni legate ad un'altra di partenza e, automaticamente, snellire il lavoro computazionale dei singoli client, poichè effettuato dal server.

4.3 Il database MySQL

Il database MySQL è stato progettato e diviso per favorire l'interoperabilità software fra le varie componenti del sistema. Due sono le tabelle principali, alle quali vengono collegate tutte le altre tramite chiave esterne[1]: *debate* e *participation*, che rappresentano rispettivamente il dibattito e un determinato intervento/partecipazione.

Ogni dato analizzato dal server in backend, con tutte le sue componenti di analisi del testo e qualsiasi configurazione testuale e d'interfaccia effettuata, viene memorizzato nel database relazione, in modo tale che i client in ascolto, o i pannelli di controllo in caso di modifica delle informazioni, possano effettuare richieste specifiche su intere tabelle o colonne, intersecando dati già in possesso o richiedendone di nuovi quando necessario. Ad esempio, quando il client di visualizzazione viene lanciato e si seleziona un determinato dibattito, la prima query effettuata è quella di richiesta delle informazioni di base della trassmissione (nome, data, tipologia etc). Una volta recuperati questi dati, tra cui l'intero identificativo univoco del dibattito, si passa a ricavare tutte le informazioni giù salvate correlate alla trasmissione d'interesse come, ad esempio, un'eventuale configurazione dei grafici, se già esistente.

[1]Vincolo di integrità referenziale tra due o più tabelle. Essa identifica una o più colonne di una tabella (referenziante) che referenzia una o più colonne di un'altra tabella (referenziata).

4.3.1 La struttura relazionale

La struttura relazionale del database, vista la natura strettamente correlata delle informazioni trattate e la loro quantità, risulta complessa e ricca di legami esterni[1] e vincoli di modifica e cancellazione[2] tra le varie tuple[3] delle diverse tabelle.

[1]Sono dei legami tra le tabelle del database.

[2]I vincoli sono dei controlli che vengono implementati in una o più colonne di una tabella.

[3]Una tupla è un generico elemento di una relazione con attributi in un database relazionale.

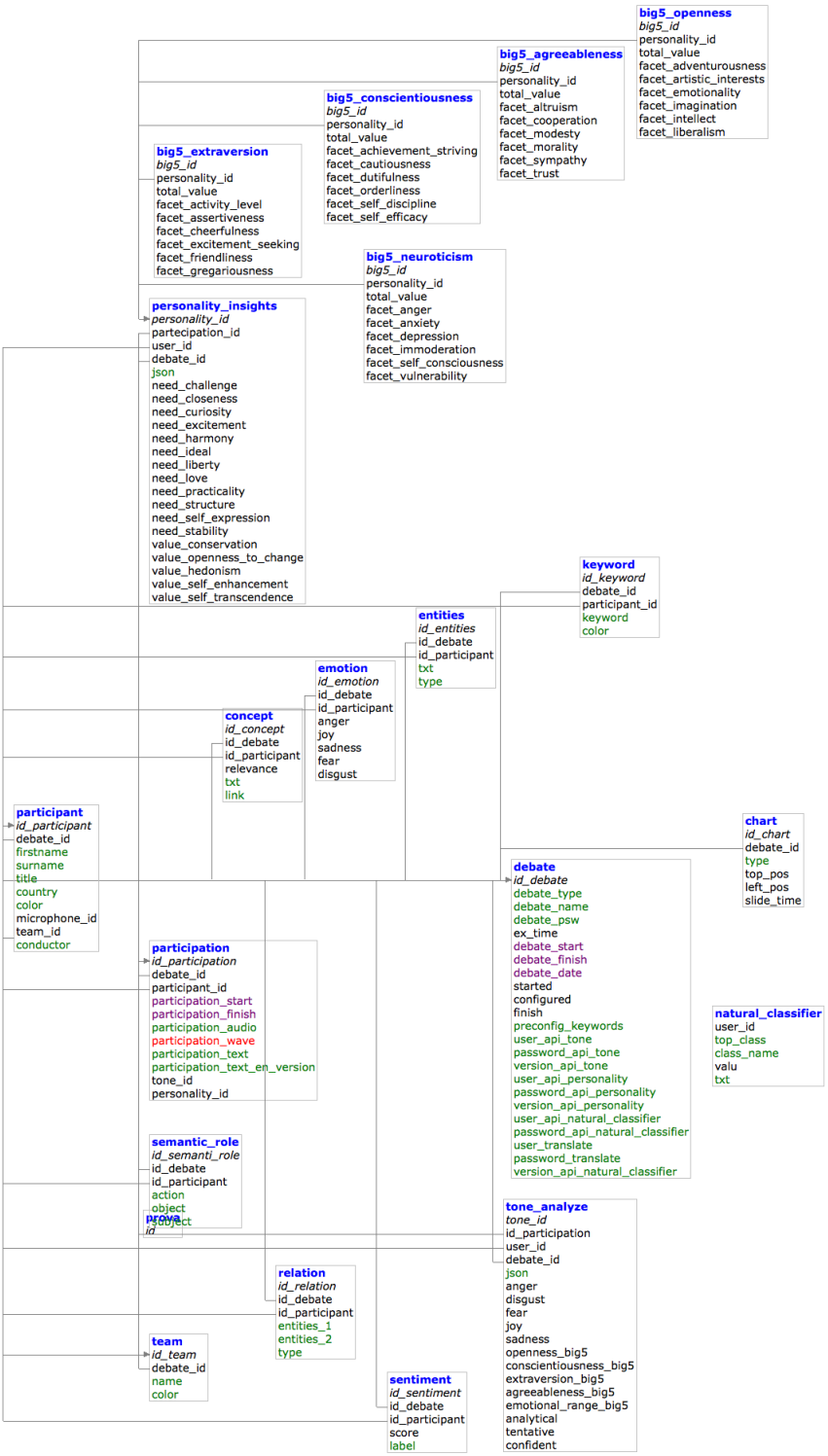

Figura 4.7: Schema del database relazionale MySQL.

4.3.2 Tecnologia Push e messaggistica tra le componenti

Il database viene aggiornato (nei dibattiti in diretta) in tempo reale dal server che recupera ed elabora l'audio estrapolandone il testo da analizzare. Quando la procedura risulta conclusa e i dati sono stati salvati, il server informa, con un messaggio di aggiornamento, tutti i client connessi in ascolto di messaggi in arrivo sul canale di comunicazione del dibattito, che è stata aggiunta o modificata qualche tupla nelle tabelle della trasmissione.

Fatto ciò, è compito dei client aggiornare i dati locali precedentemente scaricati e visualizzare, qualora necessario, le nuove informazioni prelevate nei grafici presenti.

Indipendentemente dal tipo di dibattito, tutti gli accessi al database vengono sincronizzati tramite notifiche push[1], i cui messaggi vengono generati e gestiti tramite la libreria Pusher[2], che consente una perfetta integrazione del codice basato su architettura web socket [6] nei diversi linguaggi utilizzati per l'applicazione web ed il server di gestione ed analisi dell'audio scritto in Phyton.

[1] La notifica push è una tipologia di messaggistica istantanea con la quale il messaggio perviene al destinatario senza che questo debba effettuare un'operazione di scaricamento (modalità pull).

[2] `https://pusher.com`.

```
def pusher(x):

    print x

    D_UPDATE = 'D_UPDATE_'+ str(x)

    pusher = Pusher(app_id="285847", key="b51f0e8a714083bc0879", secret="e83de8ca0e9bbd3489d2", cluster="eu")
    pusher.trigger(['my-channel'], 'my_event', {'message': D_UPDATE})
```

Figura 4.8: Esempio dell'invio di un messaggio di update dal server agli 'n' client connessi tramite Python.

Come già detto, i client di visualizzazione restano in ascolto solamente dei messaggi in arrivo dal canale di comunicazione relativo al dibattito scelto creato, in automatico dal sistema di gestione, nella fase di configurazione preliminare.

Inoltre, questa libreria mette a disposizione degli utili strumenti di controllo del flusso dei messaggi tra i vari moduli del softaware, ad esempio dei contatatori di numero delle connessioni e di messaggi scambiati, come si può vedere in Figura 4.8.

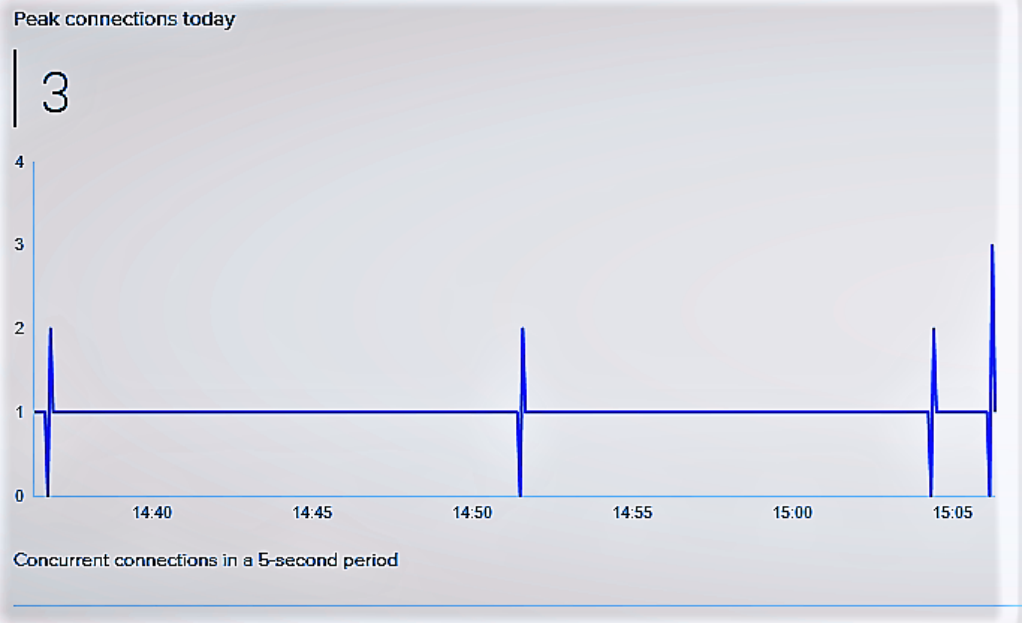

Figura 4.9: Alcuni strumenti di controllo del flusso delle connessioni e dei messaggi della libreria Pusher.

Si è preferita una tecnologia di tipo push [7], rispetto ad una di tipo pull, poichè un determinato messaggio deve essere mandato da un unico server a molti client in ascolto che, in qualsiasi momento, reagiranno di conseguenza. Inoltre, visto che l'interfaccia utente non conosce i tempi di aggiornamento o quando richiedere delle risorse al database, ma è il software sottostante che gestisce e comunica i cambiamenti nei dati da rappresentare, una messaggistica di tipo pull sarebbe stata inappropiata, proprio a causa della necessità, da parte dei client connessi, di richiedere una determinata risorsa al tempo d'esecuzione corretto. Come già detto, la maggior parte delle componenti dell'architettura devono inoltre comunicare tra loro per sincronizzare l'invio e la richiesta di alcune informazioni: questo è possibile sfruttando la combinazione di classiche query di controllo di dati PHP e

l'utilizzo di messaggi basati su tecnologia web socket, che fornisce canali di comunicazione full-duplex attraverso una singola connessione TCP [8], standardizzata dal W3C[1].

4.4 Scalable Vector Graphics SVG

Per la creazione della grafica a run-time si è scelto di utilizzare la tecnologia SVG, acronimo di Scalable Vector Graphics, disponibile con HTML5 come standard W3C. A differenza della più comune tecnologia Canvas di HTML5, che prevede la creazione di immagini statiche con dimensione prefissata non scalabili (senza perdere di qualità), SVG consente la creazione di oggetti dinamici: infatti, ogni singolo elemento disegnato a schermo viene considerato dal browser web come un oggetto a se stante che può essere modificato, in qualsiasi sua caratteristica, senza la necessità di aggiornare nuovamente l'intera pagina web. L'utilizzo della tecnologia canvas non avrebbe permesso la dinamicità richiesta né in fase di configurazione della trasmissione né in fase di visualizzazione delle statistiche, visto che ogni modifica ad una grafica di questo tipo richiede necessariamente il redraw dell'elemento radice per essere visualizzata. Lo standard SVG ha garantito, invece, la maggiore flessibilità possibile in fase di aggiornamento dei grafici ed in fase di resizing della finestra del browser: infatti, utilizzando un sistema di disegno basato sulla dimensione in percentuale della pagina, l'aggiornamento della grafica risulta immediato e preciso.

[1] `https://www.w3.org.`

```
//Update single graph
function draw_timeline_graphic(id,time){
    var speakers = document.getElementsByClassName("time_"+id);

    var line_width = $('.timeline_line')[0].getBoundingClientRect().width;;
    var parentWidth = $(window).width();
    var percent = (100*line_width)/parentWidth;

    participation_offset = (percent*time_run)/debate_sec;

    var u = (percent*time)/debate_sec;
    var nWidth = u.roundTo(5);
    speakers[0].setAttribute("width", nWidth.toString()+"%");
}
```

Figura 4.10: Funzione per l'aggiornamento delle dimensione delle singole componenti del grafico timeline basato su calcoli percentuali.

Tra l'altro, il sistema procedurale di creazione dei grafici basato su questa tecnologia vettoriale ha consentito di ottimizzare le prestazioni a tempo di esecuzione con i moderni browser, di eliminare qualsiasi tipo di immagine raster[1] così da diminuire lo spazio occupato su server e ottimizzare i tempi di caricamento: infatti l'intera infrastruttura software, comprensiva dei due client di configurazione e di quello di visualizzazione, non supera 1,5 MB.

4.5 Visualizzazione delle informazioni

Le informazioni estrapolate dal database, o quelle direttamente calcolate dal sistema, devono essere manipolate diversamente a seconda della visualizzazione richiesta. Ad esempio, in un grafico di tipo timeline, la visualizzazione degli interventi viene calcolata diversamente se il dibattito risulta termina-

[1]Nota anche come grafica bitmap, è una tecnica utilizzata per descrivere un'immagine in formato digitale contrapposta a quella vettoriale visto che le immagini vengono definite tramite una griglia di pixel.

to, attualmente in onda o non iniziato: infatti, in una trasmissione conclusa, avremo a priori il tempo di fine dell'ultimo intervento, così da poter calcolare la lunghezza totale (in percentuale) della superficie "scrivibile" dello schermo dedicata a tale grafico ed utilizzare tale informazione per poter disegnare correttamente i singoli interventi rapportati alla loro durata nel tempo.

Figura 4.11: Esempio del funzionamento della timeline.

Se invece stiamo visualizzando una trasmissione non ancora in onda, o iniziata ma non ancora conclusa, la percentuale della superficie utilizzabile dalla timeline viene gestita in base al tempo stimato per la durata del dibattito (impostato in fase di configurazione). Se tale "limite" non viene rispettato il sistema si occupa di aggiornare di volta in volta il valore fino alla conclusione della trasmissione. Oltre all'eleco dei nomi dei partecipanti, viene mostrata un'informazione relativa al "titolo" posseduto, che viene visualizzata a schermo con il colore identificativo del partecipante (sia il titolo che il colore vengono impostati dagli operatori nei rispettivi pannelli di configurazione) e, se siamo in presenza di un partecipante con il ruolo di conduttore della trasmissione, il sistema fa in modo da segnalare in maniera speciale i relativi interventi, tratteggiando la linea delle barre della timeline del conduttore. Un altro grafico interessante da analizzare è quello sull'andamento dei partecipanti, che compare in abbinamento con la timeline, e che mostra la percentuale totale di tempo parlato dai singoli sulla durata (at-

tuale) del dibattito. Similmente a come accade per la timeline, qui avremo un contatore totale del tempo trascorso e dei singoli contatori per il tempo utilizzato dai diversi partecipanti così da ottenere, con un banale rapporto matematico, le percentuali e riuscire a disegnare gli spicchi del grafico "a torta". Inoltre, potremo visualizzare a schermo il testo totale degli interventi del singolo partecipante che, in base alla lunghezza, sarà mostrato nella sua interezza nel tempo, come è possibile vedere dall'immagine di esempio sottostante.

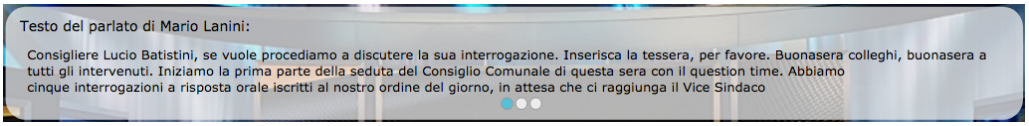

Figura 4.12: Esempio di grafico che visualizza il testo totale di tutti gli interventi di un partecipante.

Le informazioni finora visualizzate non richiedono la query di dati elaborati dai servizi di analisi del testo, ma vengono gestite in tempo reale confrontando diversi valori, come il tempo stimato, quello trascorso e quello utilizzato dal singolo partecipante, rendendo questi grafici i più complessi dal punto di vista logico. Per mostrare, invece, i risultati delle elaborazioni dei servizi di analisi di IBM, vengono utlizzati dei grafici specifici a seconda delle informazioni che si vogliono mostrare o confrontare. Ad esempio, per la visualizzazione delle statistiche sulla personalità di un partecipante o sul tono del suo discorso, si è scelto di mostrare una serie di valori percentuali calcolati sulla totalità del parlato, come visibile nei due grafici di esempio.

Figura 4.13: Esempio di grafico che visualizza le statistiche dei dati ricavati dai servizi di Tone Analyze e Personality Insights di IBM sul singolo partecipante.

Ad ogni valore, visto come proposizioni logica dal sistema, viene assegnato un grado di verità compreso tra 0 e 1 utilizzando una logica di tipo Fuzzy[1], estensione della logica booleana. I valori Fuzzy possono variare da 0 a 1 come le probabilità dei classici eventi ma, diversamente da queste, descrivono eventi che si verificano in una certa misura. Successivamente questi

[1] `https://it.wikipedia.org/wiki/Logica_fuzzy`.

valori vengono approssimati e visualizzati come percentuale a schermo. Oltre questa tipologia di grafici, le statistiche possono essere confrontate singolarmente con i relativi valori di ogni partecipante alla trasmissione: infatti, una volta disponibili tutti i valori da confrontare per un determinato dato, il sistema li visualizza in un grafico "a torta", che mostra la percentuale del singolo su quella totalizzata in totale (calcolata sui numeri fuzzy), come si può vedere in Figura 4.14.

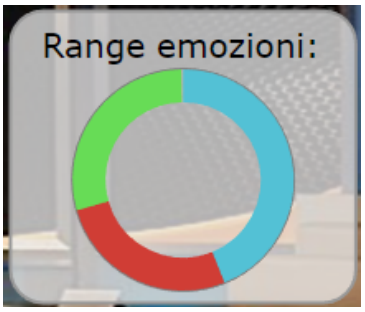

Figura 4.14: Esempio di grafico che visualizza i confronti per le statistiche dei partecipanti: si nota come il range emozionale del partecipante contrassegnato con il colore azzurro sia maggiore degli altri.

Tale tipologia di grafico è disponibile, in trenta varianti, per i valori di personalità e del tono tenuto ricavati dai servizi di analisi del testo esterni di IBM. Quando abbiamo un ventaglio di dati parziali e non visualizzabili o confrontabili, viene visualizzata un'icona che informa l'utente che i dati richiesti non sono momentaneamente disponibili.

Oltre ad informazioni statistiche su emozioni e tono della conversazione, i servizi di Natural Language Understanding[1] di IBM del testo mettono a

[1] `https://www.ibm.com/watson/developercloud/natural-language-understanding.html`.

disposizione dati estratti dal contenuto del testo, come i concetti, le entità, relazioni e significati semantici degli interventi. Tutte queste informazioni vengono rappresentate nei grafici in forma testuale, come si può vedere in Figura 4.15.

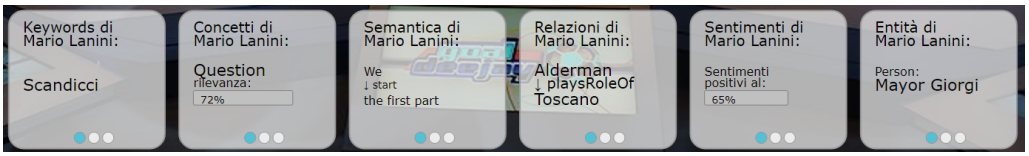

Figura 4.15: Esempio di grafici ricavati dal servizio Natural Language Understanding di IBM.

Ricapitolando, in fase di configurazione visuale della trasmissione in programma, possiamo scegliere tra dodici tipologie di grafici diverse, due delle quali hanno, rispettivamente, tredici e diciassette diversi dati da poter confrontare tra tutti i partecipanti.

4.6 Strumenti di sviluppo utilizzati

Per questo progetto ho utilizzato tutti software di sviluppo e grafica open source[1] disponibili liberamente in rete. Come IDE[2] per lo sviluppo delle applicazioni web è stato scelto Aptana Studio[3], basato su il più diffuso ambiente di sviluppo al mondo Eclipse[4]. Una tra le caratteristiche più utili di

[1]Software non protetto da copyright e liberamente modificabile dagli utenti.

[2]Ambiente di sviluppo integrato, dall'inglese integrated development environment, è il software che aiuta i programmatori nella scrittura del codice sorgente.

[3]`http://www.aptana.com`.

[4]`https://www.eclipse.org`.

questo IDE è l'aiuto nella creazione di codice HTML [1] compatibile con le ultime specifiche HTML5[1] e l'assistenza alla scrittura di CSS [2], JavaScript[2], e PHP[3] supportato dalla maggioranza dei browser[4] a disposizione come si può vedere dalla figura sottostante.

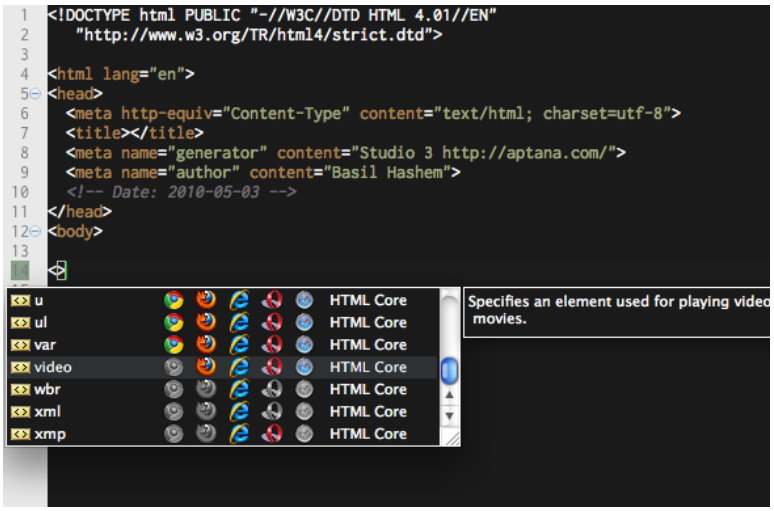

Figura 4.16: Esempio di aiuti per la scrittura di codice compatibile con i diversi browser forniti da Aptana Studio.

In fase di debugging[5] e testing, inoltre, fornisce dei comodi strumenti per settare breakpoints[6], controllare variabili ed il flusso dell'esecuzione del programma. Grazie ad un'interfaccia semplice ed intuitiva questo IDE ha

[1] `https://www.w3.org/TR/html5/`.

[2] `https://www.javascript.com/`.

[3] `http://php.net/`.

[4] L'applicazione per il recupero, la presentazione e la navigazione di risorse sul web.

[5] L'attività che consiste nell'individuazione e correzione dei problemi.

[6] Un breakpoint è un punto di interruzione nel codice di un programma, normalmente usato per scopi di debugging.

semplificato molto la gestione dell'alto numero di righe del codice scritto con diversi linguaggi per il web, sia lato client che lato server utilizzati.

Durante lo sviluppo ho usufruito, soprattutto nelle primissime fasi nelle quali il database era in via di definizione, di un server locale inizializzato in XAMPP[1], contenente MySQL[2], per testare le funzionalità della parte online dell'infrastruttura software e correggere le criticità nel codice PHP. Successivamente, per integrare il lavoro di analisi dei dati svolto dal sistema di backend tramite server[3] programmato in Python[4], ho integrato il database fornita dal server MySQL dell'Università.

Per quanto riguarda la creazione di mockup[5], di grafica di prova per le varie interfacce e di qualsiasi altro tipo di immagini, come si può vedere in figura, ho utilizzato l'editor di grafica vettoriale[6] open source Inkscape[7], risultato, in questo contesto, un potente e flessibile strumento di progettazione e di realizzazione.

---

[1] `https://www.apachefriends.org/it/index.html`.

[2] `https://dev.mysql.com/`.

[3] Componente di elaborazione e gestione del traffico che fornisce, sia a livello logico che fisico, un servizio alle altre componenti.

[4] `https://www.python.it/`.

[5] Il mockup è l'attività di riprodurre un oggetto, un modello o un'interfaccia grafica in scala ridotta o maggiorata.

[6] Nella grafica vettoriale un'immagine è descritta mediante un insieme di primitive geometriche che definiscono punti, linee, curve e poligoni.

[7] `https://inkscape.org/it`.

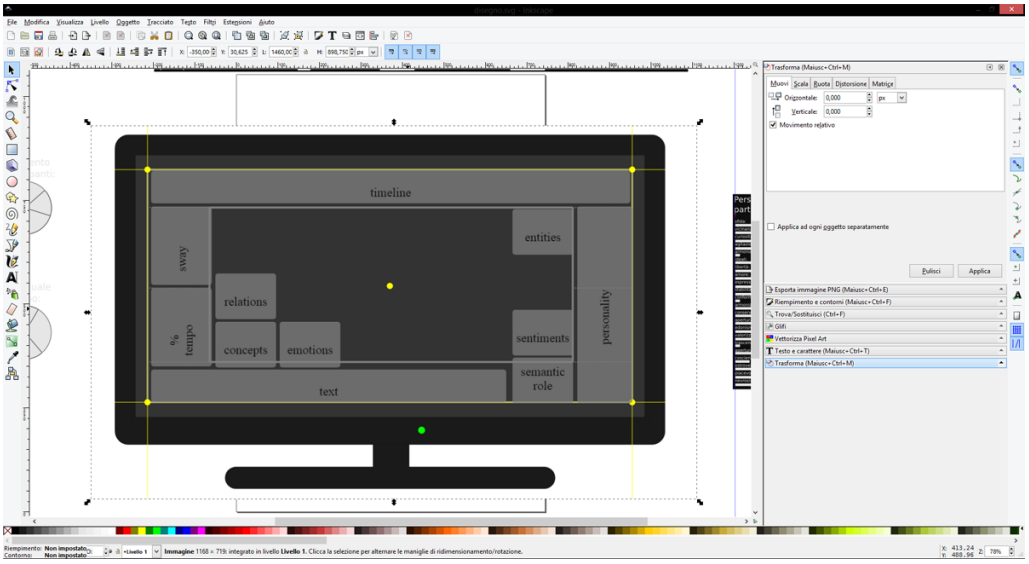

Figura 4.17: L'interfaccia semplice ed intuitiva dell'editor grafico Inkscape.

Utilizzare questa tipologia di editor di grafica, basato sulla stessa tecnologia vettoriale del software di visualizzazione di dibattito, mi ha permesso di capire e sfruttare meglio le caratteristiche di questa metodologia di disegno.

Infine i test sono stati eseguiti su tutti i principali browser disponibili e su diverse tipologie di elaboratori, in modo tale da realizzare software perfettamente compatibile e senza perdita di funzionalità sui vari dispositivi che l'utente finale potrebbe utilizzare.

Capitolo 5

Conclusioni e sviluppi futuri

In questo lavoro è stato sviluppato un sistema in grado di creare, configurare e gestire una trasmissione televisiva, di elaborare le informazioni estrapolate dai file audio dei diversi interventi e di far visualizzare a schermo ed in modo intuitivo i risultati, tramite un'apposita interfaccia grafica, agli spettatori del dibattito.

Abbiamo iniziato questo progetto di tesi con una visita in sala di registrazione, esperienza che ci ha permesso di capire le logica televisiva e di ideare l'architettura hardware e software dell'infrastruttura completa.

Lo studio dei servizi di analisi del testo esterni designati dopo un'attenta analisi e quelli effettuati per trovare le differenze tra le varie tipologie di trasmissioni hanno consentito di collezionare un alto numero di dati riconducibili agli interventi di un dibattito e, successivamente, la progettazione di grafici con i quali visualizzare le informazioni ha permesso di avere un quadro completo delle caratteristiche che i diversi moduli dell'infrastruttura software avrebbero dovuto avere.

Una volta determinato il flusso dei dati si è passati alla progettazione

di un database che potesse immagazzinare nel modo più efficiente possibile i dati prelevati dalla sala di registrazione, quelli frutto di inserimento da parte degli operatori e di elaborazione da parte del server.

Parallelamente allo sviluppo del codice per l'estrapolazione dell'audio dalle sorgenti audio e all'implementazione dei servizi di analisi esterni, è iniziato lo sviluppo dei vari moduli gestionali e di visualizzazione del sistema e, in diverse fasi, l'integrazione della messaggistica tra le due diverse parti del progetto, iniziando così i primi test funzionali svolti, in principio, su input predefiniti e successivamente simulando una vera e propria trasmissione tramite uso dell'hardware dedicato.

Tra le criticità del progetto emerse in corso d'opera la più difficoltosa da gestire è risultata sicuramente la sincronia dei diversi moduli dell'architattura nel caso di una trasmissione con elaborazione e visualizzazione dei dati in tempo reale: vista la mole di informazioni analizzate, sono risultate fondamentali, al fine di evitare ritardi e rallentamenti nel flusso dei programmi, diverse fasi di ottimizzazione del database e delle relazioni tra le tabelle per rendere le query di salvataggio e prelevamento delle informazioni più efficienti possibile. Inoltre, un'altra difficoltà degna di nota, si è presentata nella gestione di alcuni grafici, ad esempio per la timeline in una trasmissione in tempo reale, richiedendo un grande sforzo sul miglioramento della sincronizzazione della messaggistica tra le due componenti del progetto.

Su questa infrastruttura è possibile implementare altre tipologie di funzionalità e lavorare ancora su molti aspetti. Ad esempio, potrebbero essere implementate altre tipologie di analisi con relativi grafici di visualizzazione dei risultati oppure, l'intero sistema, potrebbe essere adattato per consentire

l'elaborazione di dati estrapolati da campagne di web marketing[1] sui social network al fine di comprendere le reazioni che una determinata strategia di mercato ha sui possibili clienti.

Inoltre, vista la natura totalmente multipiattaforma del codice sorgente e la grande compatibilità delle tecnologie utilizzate con la maggior parte dei dispositivi in commercio, è possibile un'integrazione con tutti i mezzi di comunicazione esistenti come, ad esempio, riproduttori di filmati eseguibili via browser o le più moderne smart tv[2].

In conclusione, un progetto del genere, già altamente innovativo e personalizzabile ad occorrenza, apre alla possibilità di innumerevoli sviluppi futuri e applicazioni in campi diversi da quello relativo alle trasmissioni televisive trattato.

[1] L'insieme delle attività di marketing che sfruttano il canale Web, come ad esempio la pubblicizzazione di un'azienda o prodotto sui social network.

[2] Apparecchiature che hanno come principale caratteristica l'integrazione di funzioni e di servizi legati a Internet.

Elenco delle figure

Bibliografia

[1] P. H. I. J. Hildebrand, Mozilla, “HTML Format for RFCs.” RFC 7992, Dec. 2016.

[2] H. Flanagan, “Cascading Style Sheets (CSS) Requirements for RFCs.” RFC 7993, Dec. 2016.

[3] R. Fielding and J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.” RFC 7230, June 2014.

[4] “Uniform Resource Identifier (URI): Generic Syntax.” RFC 3986, Jan. 2005.

[5] “The JavaScript Object Notation (JSON) Data Interchange Format.” RFC 7159, Mar. 2014.

[6] “The WebSocket Protocol.” RFC 6455, Dec. 2011.

[7] M. Thomson, E. Damaggio, and B. Raymor, “Generic Event Delivery Using HTTP Push.” RFC 8030, Dec. 2016.

[8] “Transmission Control Protocol.” RFC 793, Sept. 1981.

Ringraziamenti