Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea Specialistica in Informatica

L'UTILIZZO DI SOUNDMAP COME STRUMENTO DI MONITORAGGIO E DI PROGRAMMAZIONE PER LA TUTELA DEL TERRITORIO

Relatore:
Chiar.mo Prof.
Silvio Relandini

Presentata da:
Michele Malatesta

Sessione III
Anno Accademico 2009 - 2010

Indice

Elenco delle figure

Capitolo 1

Introduzione

Questa tesi di Informatica Musicale si basa su un progetto di ecologia sonora, una disciplina nata alla fine degli anni 60 in Canada grazie al lavoro svolto dal professor Murray Schafer, inizialmente con l'obiettivo di creare una raccolta dei suoni antropici e naturali che rischiavano di estinguersi a causa del rapido mutamento della società. Questo lavoro iniziale suscitò l'interesse di molte persone che contribuirono a realizzare migliaia di file audio campionati e registrati in ogni di parte del mondo. Questi file audio riproducevano suoni prodotti nei villaggi, in Europa, nelle Americhe e in Oceania. Negli anni 80 è nata la Natural Sound Science, una disciplina che ha utilizzato queste informazioni per studiare il livello di inquinamento sonoro in grado di destabilizzare un habitat naturale. Su queste basi alcune società americane si sono già mosse da tempo per creare ad esempio tramite la Natural Society americana una mappatura sonora della California. In Europa invece non è stato fatto ancora nulla di così specifico, sarebbe quindi interessante e utile iniziare a sviluppare un progetto che vada anch'esso in questa direzione. Questa tipologia di progetti per il momento utilizzano uno strumento che è chiamato *Soundmap*, una forma di media locativo che mette in relazione un luogo e le sue rappresentazioni sonore. Questo tipo di strumento attualmente è ancora piuttosto primitivo, non fornisce infatti alcuna informazione di tipo scientifico sui rapporti di frequenza, sugli andamenti delle frequenze e sull'analisi dei dati. L'obiettivo della tesi è di arricchire la soundmap con ulteriori informazioni di analisi audio e di prevedere come sviluppo futuro la progettazione di alcuni strumenti avanzati in modo da avere a disposizione uno strumento

scientifico adeguato. Come punto di partenza dimostrativo è stata scelta la città di Trento selezionando località specifiche dove effettuare i campionamenti. Successivamente è stata interpolata una serie di dati attraverso degli strumenti di analisi che attualmente vengono utilizzati esclusivamente per un aspetto musicale. Nell'eventualità dell'avvio di un progetto di ricerca si potrebbero modificare questi strumenti di analisi e inserirli all'interno del sistema. L'avvio di un progetto di ricerca permetterebbe inoltre di avere a disposizione una serie di dati riguardo gli effetti dell'inquinamento acustico sulle varie specie e sui vari habitat. Incrociando questi dati si potrà arrivare a capire come poter intervenire per risolvere o almeno mitigare questi effetti.

Capitolo 2

Ecologia Acustica

L'ecologia acustica [11] è lo studio dei rapporti tra gli organismi viventi e il loro ambiente, è quindi lo studio degli effetti prodotti dall'ambiente acustico (o paesaggio sonoro) sulle caratteristiche fisiche e sui modelli di comportamento degli esseri che vi abitano.

Gli studi di ecologia acustica furono intrapresi per la prima volta verso la fine degli anni sessanta nella *Simon Fraser University* (Vancouver, Canada) come parte del *World Soundscape Project* [1, 2, 7] grazie a R. Murray Schafer e al suo team di ricerca. L'innovativo approccio al suono e le numerose pubblicazioni di ricerca permisero al WSP di godere in poco tempo di fama internazionale. L'obiettivo del progetto era di ottenere un rapporto equilibrato tra la comunità umana e il suo ambiente sonoro, proponendo quindi soluzioni per un paesaggio sonoro ecologicamente equilibrato. Il primo studio prodotto dal WSP fu il *"Vancouver Soundscape*", uno studio che includeva misure di livello (producendo *isobel maps*[1]), registrazioni del paesaggio sonoro e descrizioni di una serie di caratteristiche sonore. Studi successivi portarono alla pubblicazione di *"Five Village Soundscapes"*, "*European Sound Diary*" e "*Handbook for Acustic Ecology".* Tra le varie pubblicazioni il libro di Schafer "*The Tuning of the World*" rimane tuttavia il testo di ecologia acustica più semplice e conosciuto. Il progetto ha inoltre lasciato un impressionante archivio di 300 nastri audio di ambienti sonori e numerosi documenti scritti. Quando il gruppo di ricerca originale (che includeva Howard Broomfield, Bruce

[1] Un'*Isobel Map* mostra le variazioni di pressione sonora su una determinata zona, collegando i punti dove i livelli misurati sono uguali.

Davis, Peter Huse, Barry Truax, Hildegard Westerkamp, e Adam Woog) si sciolse, la distribuzione delle pubblicazioni del WSP, nonchè la manutenzione e l'espansione del suo archivio, furono portate avanti da Barry Truax e Hildegard Westerkamp. Entrambi si fecero carico dell'eredità del progetto continuando a produrre pubblicazioni, registrazioni ed effettuando corsi di Comunicazione Acustica al dipartimento di Comunicazione della *Simon Fraser University.* A poco a poco, grazie anche alle conferenze di Schafer in molte parti del mondo, il lavoro del WSP si espanse a livello internazionale.

Dopo lo studio di questi pionieri l'interesse verso la disciplina crebbe sempre più portando nel 1993 alla creazione del *World Forum of Acoustic Ecology,* un'associazione internazionale di organizzazioni affiliate e di individui per lo studio degli aspetti sociali, culturali ed ecologici dell'ambiente sonoro.

Dal 1993 il WFAE [3] si occupa di:

- **EDUCAZIONE** all'ascolto del paesaggio sonoro, incentivando ad un ascolto accurato per comprendere il significato dei suoni ambientali elevando quindi la consapevolezza uditiva.

- **RICERCA E STUDIO** di tutti gli aspetti del paesaggio sonoro. Alcuni esempi sono:

    1. il monitoraggio e la valutazione di azioni che possono alterare la qualità dell'ambiente sonoro, studiando gli effetti della tecnologia sull'ambiente acustico;
    2. l'importanza dei mezzi di comunicazione elettro-acustica (radio, TV ecc.), sempre più presenti nel panorama sonoro;
    3. lo studio del silenzio visto nel contesto delle differenti culture.

- **PUBBLICAZIONE E DISTRIBUZIONE** di informazioni e ricerche sull'ecologia acustica.

- **PROGETTAZIONE E CREAZIONE** di ambienti sonori sani e acusticamente equilibrati.

Dalla creazione del WFAE a Banff (Canada) è seguita quella di un simposio internazionale con cadenza triennale; in uno di questi incontri [4] si è discusso sul concetto di

ecologia acustica, su cos'è o cosa dovrebbe essere. Gran Sonnex (UK) ha proposto una classificazione secondo quattro aspetti:

1. campagne o "crociate", per identificare e ridurre al minimo l'inquinamento acustico e conservare i suoni caratteristici di ambienti o luoghi particolari;

2. "scientifico" e accademico, per studiare come gli esseri umani e gli altri organismi si relazionano al suono nell'ambiente;

3. educativo, per incoraggiare a prendere coscienza del suono ambientale;

4. artistico, attraverso composizioni di paesaggi sonori (opere a nastro che si avvalgono del suono registrato per evocare un particolare ambiente).

2.1 Paesaggio sonoro

Il concetto di paesaggio sonoro [8, 10] è stato introdotto da R. Murray Schafer nel suo libro del 1977 "*The Tuning of the World*". Per Schafer la definizione di paesaggio sonoro include tutti i suoni di un particolare ambiente che raggiungono l'orecchio umano. Del paesaggio sonoro possono far parte:

- i suoni dell'ambiente acustico naturale, costituito dai suoni naturali (tra cui ad esempio le vocalizzazioni degli animali e i suoni degli elementi naturali);

- i suoni ambientali creati dall'uomo quali la composizione musicale, il *sound design*, altre ordinarie attività come la conversazione e il lavoro, suoni di origine meccanica derivanti dall'utilizzo della tecnologia industriale.

La perturbazione di questi ambienti acustici dà luogo all'inquinamento acustico.

Secondo Schafer [11] un paesaggio sonoro ha tre elementi principali:

- Tonica (*Keynote Sound)*: nella musica, la tonica indentifica la chiave o la tonalità di una determinata composizione. Rappresenta il tono fondamentale attorno al quale può muoversi una composizione e in rapporto alla quale anche le altre tonalità acquistano significato. Negli studi sul paesaggio sonoro le toniche sono quei suoni

che vengono percepiti, in una data società, di continuo o con tale frequenza da costituire uno sfondo sul quale vengono poi percepiti gli altri suoni, ad esempio il rumore del mare per una comunità che viva accanto ad esso, o il rumore del motore a combustione interna per le città moderne. Spesso queste toniche non vengono percepite in maniera consapevole, ma nondimeno condizionano la percezione degli altri suoni e dei segnali. Nel rapporto figura/sfondo che troviamo nel campo della percezione visiva esse rappresentano lo sfondo.

- Segnale (*Sound Signal*): qualsiasi suono verso cui si rivolga l'attenzione in modo particolare. Negli studi relativi al paesaggio sonoro, i segnali si distinguono dalle toniche nello stesso modo in cui nel campo della percezione visiva si contrappongono figura e sfondo.

- Impronta sonora (*Soundmark)*: il termine deriva da *landmark* (punto di riferimento, pietra miliare) e si applica a quei suoni comunitari che sono unici o che possiedono delle peculiarità tali da far sì che gli abitanti di una comunità abbiano nei loro confronti un atteggiamento e una capacità di riconoscimento particolari.

Purtroppo, a partire dalla rivoluzione industriale, numerosi paesaggi sonori [2, 11] sono scomparsi, completamente sommersi dalla nube di rumore anonimo e omogeneo che costituisce il paesaggio sonoro della città contemporanea, con la sua onnipresente tonalità di traffico. Il contrasto tra gli ambienti acustici pre-industriali e post-industriali è ben espresso da Schafer [11] nell'uso dei termini "*hi-fi*" (alta fedeltà), per caratterizzare i primi, e "*lo-fi*" (bassa fedeltà) per descrivere questi ultimi.

- Hi-fi: abbreviazione dell'espressione inglese *High-fidelity*, alta fedeltà. Indica un rapporto segnale-rumore soddisfacente. Il termine viene utilizzato soprattutto in elettro-acustica. Applicato allo studio del paesaggio sonoro, un ambiente *hi-fi* è quell'ambiente in cui i suoni possono essere percepiti distintamente, senza che vi siano affollamento o effetti di mascheramento.

- Lo-fi: abbreviazione dell'espressione inglese *Low-fidelity*, bassa fedeltà. Indica un rapporto segnale-rumore insoddisfacente. Applicato allo studio del paesaggio sonoro, un ambiente *lo-fi* è un ambiente in cui i segnali sono così numerosi da

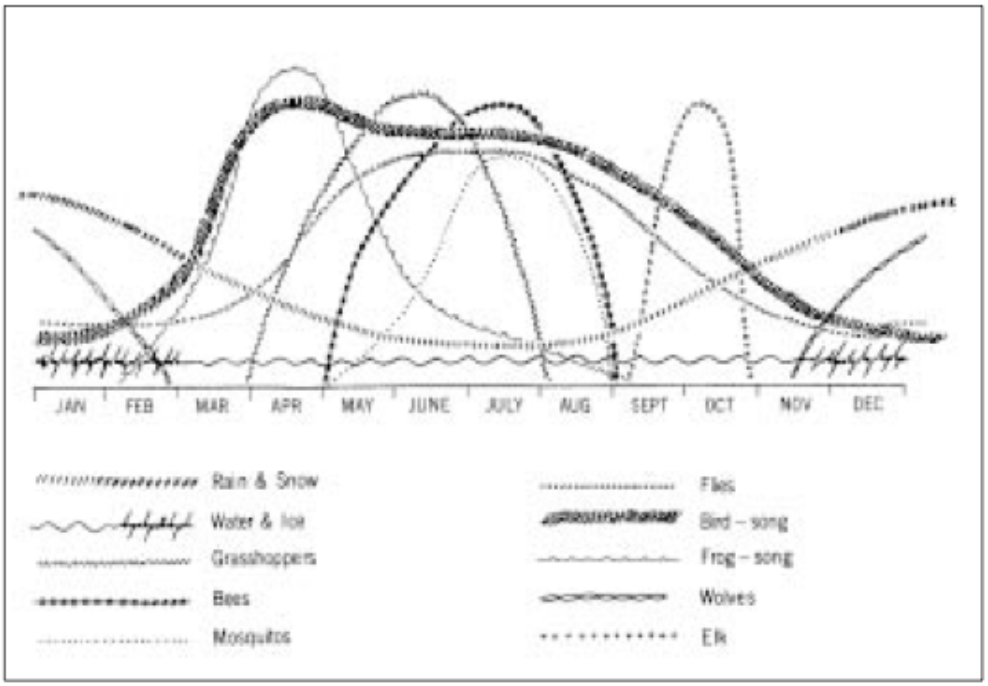

Figura 2.1: Livello dei suoni naturali [2]

sovrapporsi, con il risultato di una mancanza di chiarezza e la presenza di effetti di mascheramento.

Nella trascrizione delle registrazioni di ambienti *hi-fi*, la squadra di Schafer ha osservato che il livello dei suoni dell'ambiente naturale, come il tempo e le vocalizzazioni degli animali, varia in cicli ripetuti. Il team ha creato un diagramma rudimentale (figura 2.1) mostrando le caratteristiche più importanti del paesaggio sonoro nel corso di un periodo di dodici mesi.

Schafer ha concluso che il "dare e avere" tra le specie (evidente nella figura 2.1) è probabilmente una caratteristica dei paesaggi sonori naturali. In aggiunta al bilancio ritmico nel livello sonoro identificato da Schafer, Krause ha suggerito che in tutto lo spettro audio è evidente un equilibrio. Questa possibilità si è verificata durante i lunghi soggiorni di Krause nel deserto, nell'attimo in cui ha tentato di registrare le vocalizzazioni di creature particolari. Ascoltando con attenzione il paesaggio sonoro per catturare suoni specifici (spesso attendendo anche per 30 ore in una sola seduta), Krause ha notato che

quando vocalizzava un uccello, un mammifero o un anfibio, le voci sembravano adattarsi, in termini di frequenza e prosodia (ritmo), a tutti i suoni naturali.

Le mappe acustiche spettrografiche trascritte da 2.500 ore di registrazioni hanno confermato i suoi sospetti: le vocalizzazioni di animali e insetti tendevano ad occupare piccole bande di frequenza lasciando "nicchie spettrali" (bande di piccola o nessuna energia) in cui si inserivano le vocalizzazioni (fondamentali e formanti) degli altri animali, come uccelli o insetti. La diffusione delle aree urbane ha creato un pericolo per questi ambienti sonori. Il rumore che le accompagna infatti potrebbe ”bloccare” o ”mascherare” nicchie spettrali e, se le chiamate di accoppiamento diventassero inascoltate, una specie potrebbe estinguersi. Un recente studio della Royal Society per la protezione degli uccelli ha dimostrato che gli uccelli che vivono vicino alle strade non potendo sentirsi l'un l'altro non possono comunicare con i loro potenziali *partner*.

In un paesaggio sonoro *hi-fi* è possibile sentire distintamente tutti i suoni di tutte le frequenze in quanto non è presente il fenomeno del mascheramento (questo perchè i suoni di basso volume in genere non si mascherano l'un l'altro a meno che le loro frequenze non siano vicine tra loro). La mancanza di mascheramento facilita la propagazione della "colorazione acustica" causata dagli echi e dai riverberi (che si verificano nell'attimo in cui il suono viene assorbito e riflesso dalle superfici all'interno dell'ambiente) e dall'effetto di fattori legati alle condizioni atmosferiche, quali temperatura, vento e umidità. La colorazione risultante offre informazioni significative all'ascoltatore, fornendo spunti relativi sulla natura fisica del contesto ed esprimendo la propria dimensione in relazione a chi ascolta.

Un'altra caratteristica del paesaggio sonoro *hi-fi* è che "l'orizzonte acustico" può estendersi per molti chilometri (i suoni provenienti dalla comunità di un ascoltatore possono essere sentiti a notevole distanza). Quando è possibile ascoltare i suoni provenienti da insediamenti adiacenti diventa possibile stabilire e mantenere rapporti con le comunità locali. Nel paesaggio sonoro *lo-fi* i suoni significativi (e qualsiasi colorazione acustica associata) possono essere mascherati riducendo lo "spazio sonoro" di un individuo in maniera drammatica.

Se il mascheramento dei suoni riflessi è così grave che un singolo individuo non può sentire nemmeno il suono dei suoi passi (cosa comune per le strade di molte città) lo

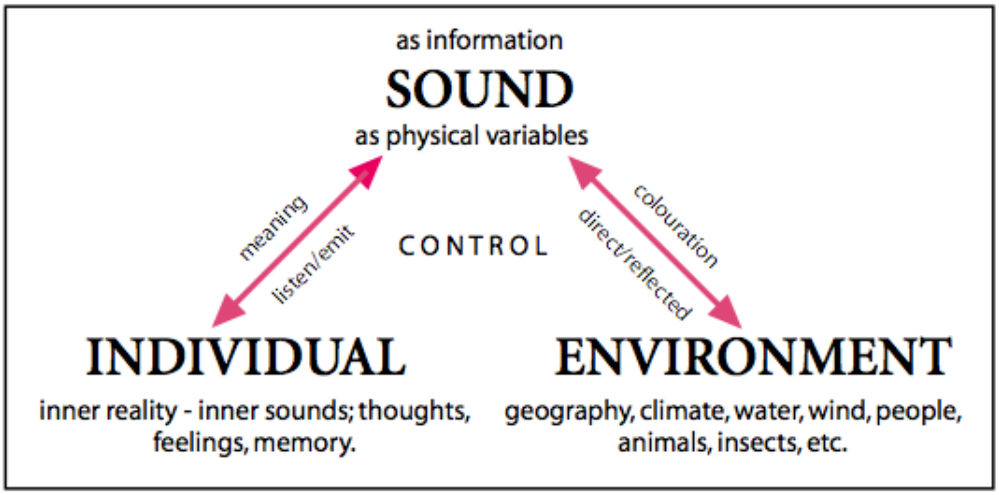

Figura 2.2: Suono come mediatore tra ascoltatore e ambiente [2]

spazio sonoro di una persona diventa estremamente ridotto. In tali condizioni estreme il suono può essere:

- soffocato (nel senso che i suoni particolari non si sentono)
- fuso con altri suoni, trasformandosi in ”rumore”.

Mentre il paesaggio sonoro *hi-fi* è equilibrato in termini di livello, spettri e ritmo, il paesaggio sonoro *lo-fi* è dotato di un livello quasi costante. Questo crea un "*SoundWall*" (muro sonoro) che isola l'ascoltatore dall'ambiente. Spettralmente il paesaggio sonoro contemporaneo *lo-fi* è sbilanciato verso la gamma di frequenze basse (grazie al motore a combustione interna e ai suoni connessi all'energia elettrica).

2.1.1 Il Paesaggio sonoro e la società

Nel descrivere la capacità del paesaggio sonoro [2] di trasmettere informazioni, Truant descrive il suono come un mediatore tra l'ascoltatore e l'ambiente. Questo rapporto è illustrato in figura 2.2.

Nel momento in cui il paesaggio sonoro si deteriora la consapevolezza delle sottigliezze del suono ambientale si inaridisce in proporzione. Come risultato i suoni significativi dei paesaggi sonori contemporanei tendono ad essere polarizzati dall'ascoltatore in due

estremi: ”forte” e ”silenzioso”; ”notato” o ”inosservato”; ”bello” (mi piace) o ”brutto” (non mi piace). La nostra società predilige la modalità visiva a quella uditiva. È interessante confrontare il nostro livello di consapevolezza sonora con quello degli uomini Kaluli di Papua (Nuova Guinea) che sono in grado di imitare il suono di almeno 100 uccelli, ma che hanno difficoltà a fornire informazioni descrittive visive. In altre parole, i suoni ambientali per la tribù Kaluli comprendono un *continuum* che offre una gamma infinita di sfumature.

Nella nostra società il suono ha meno importanza e l'opportunità di sperimentare suoni "naturali" diminuisce sempre più di generazione in generazione a causa della distruzione degli habitat naturali. Il suono diventa quel qualcosa che l'individuo cerca di bloccare invece che ascoltare. Il *lo-fi*, il paesaggio sonoro con poche informazioni, non ha nulla da offrire. Come risultato molte persone cercano di bloccare il suono esterno attraverso l'utilizzo di doppi vetri o con il profumo acustico, la musica. La musica è in questo contesto usata come mezzo per controllare l'ambiente sonoro piuttosto che come una naturale espressione artistica.

La rete, i trasmettitori e i satelliti estendono la comunità acustica attraverso l'intero pianeta. Schafer definisce quest'ultimo uso del suono come ”imperialismo sonoro”. Nel 1993 nel Regno Unito un'indagine sugli atteggiamenti pubblici riguardo al rumore elenca come principale fonte di irritazione i rumori prodotti dai vicini di casa, dalle fonti di trasmissione e dai suoni registrati (che Schafer chiama suono "Schizofonico" [2]).

Il suono può essere utilizzato come forza di controllo, come arma (per attaccare) o come barriera (per difendersi) contro il paesaggio sonoro. Psicologicamente il significato di ciò è che l'ambiente e la comunità sono diventati nemici. Come in ogni guerra, l'ambiente diventa un campo di battaglia e soffre tanto quanto i suoi abitanti.

Schafer ha stimato che la battaglia tra l'espressione sonora e il controllo contribuisce ad aumentare i livelli sonori ambientali da 0,5 a 1 dB l'anno, diventando quindi un vero e proprio generatore di rumore (figura 2.3).

[2] SCHIZOFONIA [11] : derivante dal greco *schizo*, separazione e *phonè*, voce, suono. Questo termine indica la frattura esistente tra un suono originale e la sua riproduzione elettroacustica.

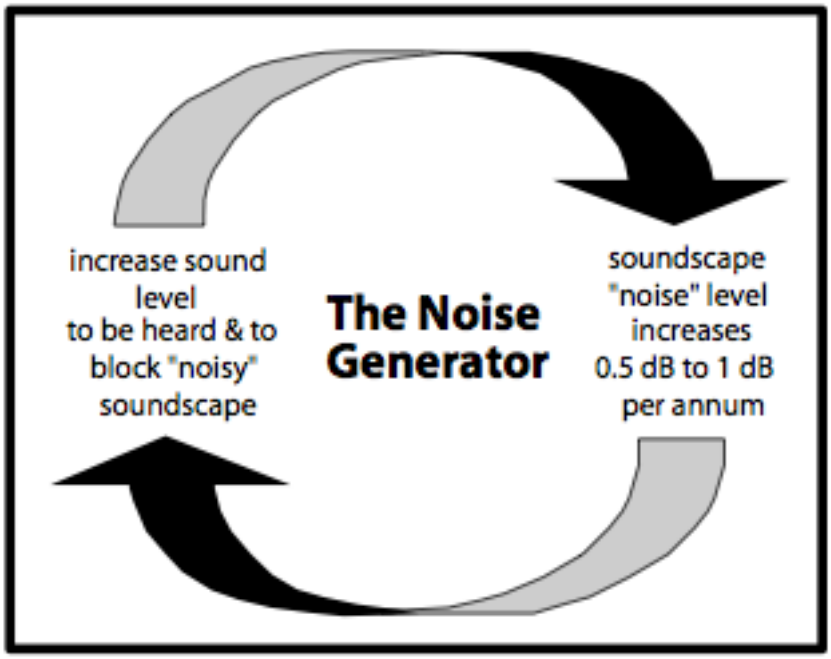

Figura 2.3: Generatore di rumore [2]

2.1.2 Migliorare il paesaggio sonoro

Per il WSP [7] produrre suoni è correlato all'ascolto. Questa relazione diventa sbilanciata se, per esempio, quello che ascoltiamo è più forte dei suoni che produciamo, oppure se l'atmosfera di un ambiente ci permette solo di ascoltare, ma non di parlare o di esprimerci. Un ambiente rumoroso annega i nostri passi, il nostro respiro, la nostra stessa voce.

Aspetto educativo

Il paesaggio sonoro rispecchia le condizioni sociali, tecnologiche e naturali del territorio. Uno dei compiti principali dell'ecologia sonora è quello di mantenere un ambiente acustico equilibrato. Secondo il WSP la strategia più importante è l'educazione. Aumentare la consapevolezza dello stato attuale del paesaggio sonoro attraverso esercizi di ascolto attivo e di "pulizia dell'orecchio [3]" è uno dei maggiori punti di forza del gruppo.

[3]Per pulizia dell'orecchio si intende [11] un programma sistematico per esercitare l'orecchio ad ascoltare con maggiore capacità di discriminazione i suoni, soprattutto quelli ambientali.

Schafer [2] ha adottato come punto di partenza la constatazione dell'incredibile predominanza della modalità visiva nella società, la cosiddetta "cultura dell'occhio", predominanza rilevabile nel deterioramento dell'abilità di ascoltare dei bambini. Schafer indirizzò il problema (che chiamò competenze sonologiche), tramite esercizi pratici che sviluppò lavorando con alcuni studenti di musica. Un tipico esercizio consisteva nell'elencare cinque suoni che piacevano e cinque che non piacevano. Come risultato molti studenti non ricordavano consciamente di aver ascoltato alcuni suoni e molti non riuscivano a completare l'elenco prima di quindici minuti. La risposta di Schafer al problema è stata di sviluppare una serie di esercizi di "pulizia dell'orecchio" includendo "passeggiate sonore", passeggiate meditative che hanno come obiettivo il mantenimento di un alto livello di coscienza sonora.

Ogni giorno spendiamo molto tempo a produrre rumore di ogni tipo [25]. Alcuni suoni prodotti hanno una valenza significativa perchè forniscono informazioni utili, altri servono semplicemente per intrattenere, altri ancora sono il risultato delle attività umane, una sorta di sfondo ambientale della nostra vita quotidiana. È importante rilevare come anche momenti di silenzio spesso siano riempiti da suoni prodotti unicamente col fine di ridurre l'ansia da "vuoto sonoro". L'ambiente acustico che ci circonda è pieno di suoni più o meno ricchi di sfumature che possono arricchire la nostra vita quotidiana o distorglierci da essa. È fondamentale acquisire un ascolto mirato in modo da poter apprezzare i paesaggi sonori che ci circondano. L'ascolto mirato consiste nel fermarsi un attimo ad ascoltare consapevolmente tutti i suoni intorno a noi. Questo è il primo passo per aprire la mente e le orecchie al paesaggio sonoro. Una semplice passeggiata può essere un'ottima occasione per concentrarsi all'ascolto del paesaggio sonoro. Bisogna però attenersi ad alcune regole: evitare di parlare, pianificare un itinerario in un paesaggio sonoro che presenti diverse varietà di suoni, in ultimo riflettere su ciò che si è sentito. Il passo successivo prevede passeggiate in posti sempre più silenziosi (chiaramente bisogna aver sviluppato una certa sensibilità all'ascolto). Il tempo dedicato alla passeggiata sonora dovrebbe essere almeno tra i trenta e i sessanta minuti. Un'ipotetica passeggiata potrebbe ad esempio prevedere un passaggio per le vie cittadine per poi giungere in un parco, potendo rilevare così vari tipi di suoni, dal rumore del traffico al cinguettio degli uccelli. Anche il rumore infatti può essere significativo, saper ascoltare consiste in un

aumento di consapevolezza, non in una battaglia contro il rumore. Il tintinnio della pioggia, le onde del mare che si infrangono sugli scogli, il calpestio dell'erba secca, il fruscio delle foglie che cadono in autunno... quando si torna da una passeggiata sonora non si è più come prima, si procede in silenzio nella scoperta della ricchezza dei suoni.

Suoni informativi

Esistono dei criteri [9] che ci permettono di asserire se un paesaggio sonoro è "sano". La descrizione di Truax è ben precisa ed è incorporata nel concetto dil hi-fi ideale di Schafer: paesaggi sonori equilibrati che spingono ad un ascolto attivo e ad un piacere sonoro. La strategia ideale è quella di massimizzare i suoni piacevoli e carichi di informazioni e di minimizzare quelli indesiderati e non informativi.

Una delle più dibattute fonti di inquinamento sonoro è il traffico. Il rumore costante in una città potrebbe non rappresentare un buon paesaggio sonoro, ma consiste realmente in un insieme di suoni non informativi? Consideriamo ad esempio il rumore del traffico mentre stiamo per addormentarci: il volume è tale da disturbarci, mentre la nostra attività richiederebbe il silenzio. Il rumore costante sembrerebbe quindi essere tutto fuorchè informativo, ma la situazione cambierebbe radicalmente se fossimo per strada o camminando in bicicletta, in questo contesto gli stessi suoni sarebbero estremamente informativi. Il suono delle macchine che circonda l'ascoltatore lo informa su quel che gli accade attorno aiutandolo ad adattarsi a condizioni in rapida evoluzione. Una macchina in avvicinamento verrebbe avvertita anche se non vista. Se le macchine non fossero rumorose questo non sarebbe possibile. Per chi si trova nel traffico il rumore è di vitale importanza, permette di interagire con gli altri ed è di sostegno alla guida (ad esempio il suono aiuta a capire quando bisogna cambiare marcia). I suoni possono aiutare nell'orientamento così il suono del traffico può indicare all'ascoltatore la direzione giusta per raggiungere una strada particolare, il che risulterebbe essere un'informazione assai valida se ci trovassimo dispersi in un bosco. A quanto pare gli stessi suoni possono essere sia informativi che non a seconda del contesto. La capacità informativa di un suono può dipendere anche dagli altri suoni presenti nello stesso paesaggio sonoro. La suoneria di un cellulare ad esempio potrebbe rivelarsi informativa se ci fossero nelle vicinanze pochi

cellulari e potrebbe non esserlo se ce ne fossero tanti e fosse difficile capire da quale apparecchio provenga.

Da non sottovalutare sono i valori estetici attribuibili al suono, ognuno ha le proprie idee su come il proprio ambiente dovrebbe essere. È difficile parlare di un criterio oggettivo, che i valori estetici varino da persona a persona infatti sono quasi tutti d'accordo. Anche se esistessero dei suoni "piacevoli" su cui tutti potrebbero concordare non è automatico che ognuno li voglia nel paesaggio sonoro che desidera.

A quanto pare l'informatività di un suono sembra dipendere dall'interazione tra gli agenti e l'ambiente, il suono più influenza l'attività dell'agente più è informativo. Per "attività" si considerano anche le azioni mentali quali pianificare, risolvere problemi, ragionare etc. Bisogna però notare che sebbene i suoni silenziosi e significativi influenzino le attività di un agente potrebbero comunque dare fastidio e quindi rappresentare un'esperienza tutt'altro che piacevole. Al contrario alcuni suoni non significativi potrebbero essere preferiti ad altri informativi (questo spiega perchè alcune persone trovano rilassanti certi tipi di rumore).

Esempi di design acustico

Un suono può essere considerato "voluto" o "non voluto" a seconda del contesto, ad esempio in un centro commerciale la proposta di miglioramento acustico potrebbe essere garantire che i suoni prodotti dalle persone (suoni voluti) non siano mascherati dalla musica e dal rumore del traffico (suoni non voluti). In una zona destinata alla riflessione un miglioramento acustico potrebbe invece richiedere di nascondere i suoni della gente, le voci diventerebbero suoni indesiderati che necessiterebbero di essere mascherati da un altro suono.

Vediamo tre esempi di design acustici per migliorare il paesaggio sonoro di una ipotetica piazza [12].

La figura 2.4 mostra una piccola piazza con una struttura d'acqua situata (come in molte piazze) vicino ad una strada.

- Scenario 1: l'intento è di creare un luogo piacevole in una zona interna della città, un posto non inquinato dai rumori. In questo ipotetico paesaggio sonoro l'acqua in movimento dovrebbe essere il suono dominante. Il suono voluto (l'acqua) dovrebbe

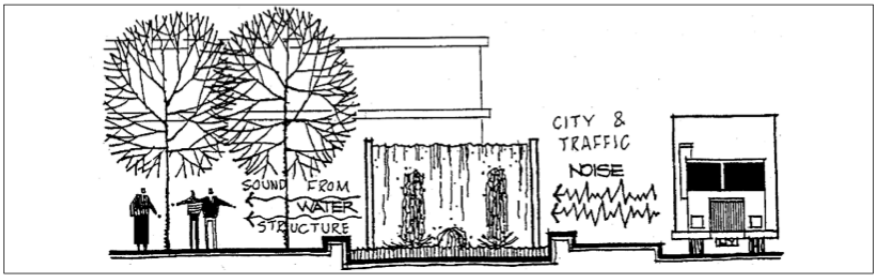

Figura 2.4: Ipotetico contesto di design urbano [12]

quindi mascherare il suono non voluto (il traffico della città). Per far questo si potrebbe ridurre il volume del rumore del traffico oppure aumentare il volume del suono della struttura d'acqua. La scelta dipende da diverse variabili, quali il contesto, i fondi economici e le possibilità tecniche.

- SCENARIO 2: si vuole creare un luogo dove si possa parlare senza che i rumori esterni coprano il suono delle voci. I suoni voluti (le voci delle persone) non dovrebbero quindi essere mascherati da quelli non voluti (traffico e struttura d'acqua). Anche in questo caso è necessario effettuare delle misurazioni di livello.

- SCENARIO 3: si dà la precedenza all'utilità che il suono del traffico (soprattutto i segnali dei veicolo) assume nei confronti di un pedone (che potrebbe essere anche un bambino o un non vedente). In questo caso il suono della struttura d'acqua diventerebbe non voluto qualora mascherasse il suono del traffico (suono voluto).

2.2 Rumore

Etimologicamente, il termine inglese *noise* può essere fatto risalire al termine dell'alto francese *noyse* e ai termini provenzali del secolo XI, *noysa, nosa, nausa*, ma la sua origine rimane comunque incerta. R. Murray Schafer sostiene che il termine possiede una varietà di significati e di sfumature, i più importanti dei quali sono i seguenti: [11]

1. SUONO NON DESIDERATO. L'*Oxford English Dictionary* contiene citazioni di *noise* inteso come "suono non desiderato" risalenti almeno fino al 1225.

2. Suono non musicale. Nel XIX secolo il fisico Hermann Helmholtz usò il termine per descrivere i suoni composti di vibrazioni aperiodiche (lo stormire delle foglie), in opposizione ai suoni musicali, composti di vibrazioni periodiche. In questo senso, il termine *noise* è oggi usato in espressioni quali "rumore bianco" o "rumore gaussiano".

3. Tutti i suoni di forte intensità. Oggi, nell'uso quotidiano, il termine viene spesso usato per indicare i suoni particolarmente forti. È rifacendosi a questa accezione del termine che certi regolamenti sul controllo e sulla riduzione del rumore proibiscono determinati suoni di forte intensità o fissano dei limiti espressi in decibel.

4. Disturbo all'interno di qualsiasi sistema di comunicazione. In elettronica e in meccanica il termine *noise* indica quasiasi disturbo che non faccia parte del segnale, come il suono prodotto dall'elettricità statica in un telefono o dall'effetto neve su uno schermo televisivo.

In termini generali, la più soddisfacente definizione di rumore è ancora oggi quella di "suono non desiderato".

2.2.1 Effetti del rumore sull'uomo

Quando ci si lamenta per il rumore spesso ci riferisce solo a suoni di alta intensità [24] ed è facile incontrare la parola decibel. Quello che spesso accade è che in risposta alla lamentela arriva una persona "addetta ai lavori" che dopo aver borbottato qualcosa sui decibel spiega che la lamentela è infondata sostenendo che il rumore non era poi così forte. Dopo aver parlato di logaritmi e impressionato con lunghi discorsi porterà a respingere il reclamo. Spesso però le argomentazioni per respingere un reclamo non hanno fondamento scientifico. Uno degli errori comuni ad esempio è considerare che un aumento di 10 decibel corrisponda solo ad un raddoppio del volume percepito. Capire quanto un suono possa essere alto può essere visto da tre punti di vista:

1. fisico;

2. fisiologico;

3. psicologico.

Approccio fisico

Come sappiamo il suono è un' onda di pressione acustica che si propaga attraverso un mezzo.

L'onda di pressione acustica che è in grado di indurre nell'uomo la sensazione sonora di più piccola intensità [23] ha una variazione di pressione di 20 µPa [4] (ovvero venti milionesimi di Pa, cinque miliardi di volte più debole della pressione atmosferica), mentre quella che induce una sensazione sonora di massima intensità ha una variazione di pressione di 20 Pa. Avendo a che fare con un milione di valori non ha molto senso utilizzare un asse lineare (figura 2.5) per rappresentare quest'intervallo e si preferisce quindi un asse logaritmico (figura 2.6).

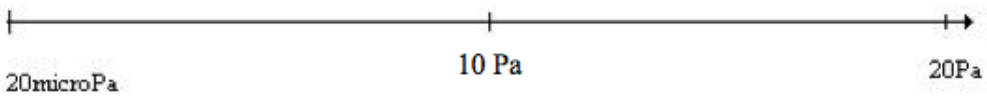

Figura 2.5: Asse lineare

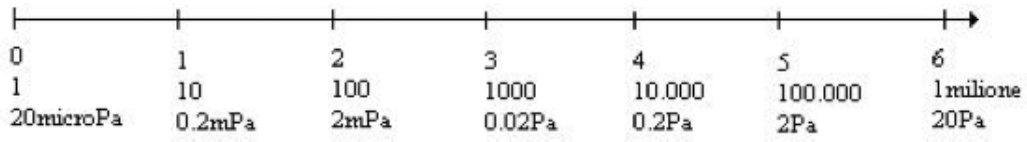

Figura 2.6: Asse logaritmico

Come si misura l'intensità della pressione sonora?

Bisogna inanzitutto precisare che quello che si fa è calcolare un rapporto, cioè il valore di un determinato suono rispetto a quello di un suono detto di riferimento, che nel nostro

[4] Il pascal è un'unità di misura utilizzata per misurare piccole variazioni di pressione. Nel caso di piccole variazioni si preferisce alla più comune *atmosfera*.
$1atm = 101.325Pa$

caso, coincide con il valore del suono di intensità più piccola : 20 µPa. Il calcolo che si effettua è il seguente:

$$\frac{P}{P_R} \; dove \; P_R = 20\mu Pa$$

Come esempio, valutiamo P = 0.2 Pa :

$$\frac{P}{P_R} = \frac{0.2Pa}{20\mu Pa} = \frac{2 \cdot 10^{-1} Pa}{2 \cdot 10^{-5} Pa} = 10.000$$

Ovvero il suono P ha un'intensità 10.000 volte maggiore di P_R e se facciamo il logaritmo di questo rapporto otteniamo:

$$log_{10} \frac{P}{P_R} = 4$$

Per migliorare la leggibilità dei valori e del grafico si moltiplica il logaritmo per un fattore 20 ottenendo così la formula del decibel (dB), uno strumento che permette di misurare l'intensità del suono (la cui unità di misura resta comunque il pascal):

$$dB = 20log_{10} \frac{P}{P_R}$$

Abbiamo così realizzato una scala di intensità della pressione sonora che va da un valore minimo di 0 dB ad un valore massimo di 120 dB (figura 2.7).

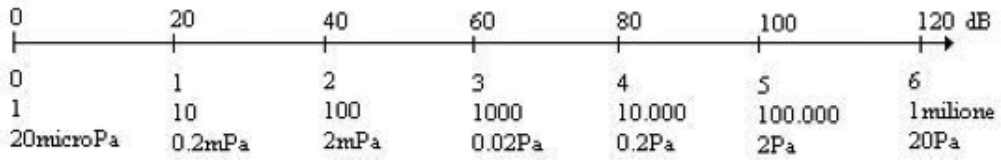

Figura 2.7: Scala dei decibel

L'intensità sonora può essere espressa anche in Watt (W) ovvero nell'unità di potenza:

$$dB = 10log_{10} \frac{W_1}{W_2}$$

considerando che la potenza è proporzionale al quadrato della pressiona sonora:

$$dB = 20log_{10}\frac{P_1}{P_2} = 10log_{10}\left(\frac{P_1}{P_2}\right)^2 = 10log_{10}\frac{W_1}{W_2}$$

i valori di riferimento sono quindi:

- per la potenza $W_2 = 10^{-12}Watt$
- per la pressione $P_2 = 2 \cdot 10^{-5}Pascal$

Per raddoppiare l'intensità di un suono è sufficiente aggiungere 6 dB. Questo è facile da verificare con un semplice esempio:
prendiamo un suono iniziale di 0.2 Pa corrispondente ad un valore di 80 dB e calcoliamo quanti dB bisogna aggiungere al segnale per arrivare ad un'intensità doppia

$$dB = 20log_{10}\frac{P_1}{P_2} = 20log_{10}\frac{0.4}{0.2} = 20log_{10}2 = 20 \cdot 0.3 = 6$$

nel caso della potenza sonora:

$$I_{dB} = 10log_{10}\frac{2W}{W} = 10log_{10}2 = 10 \cdot 0,3 = 3dB$$

ovvero l'aumento è di 3 dB.

È quindi evidente che quando qualcuno sostiene che un suono a 60 dB non è tanto più forte di uno di 55dB sta affermando (volontariamente o meno) una cosa estremamente scorretta.

Approccio fisiologico

L'orecchio umano non è ugualmente sensibile a tutte le frequenze, è infatti più sensibile ai suoni situati tra 1kHz e 4kHz rispetto ai suoni di frequenza molto bassa e molto alta. La risposta in frequenza dell'orecchio varia da individuo a individuo ed è fortemente correlata all'età e all'intensità del suono. La sensibilità relativa dell'orecchio alle diverse frequenze può essere misurata (in maniera un pò soggettiva) chiedendo ad un individuo di confrontare due suoni di frequenza diversa e variando l'intensità di un suono fin quando l'individuo non li percepisca ad un volume uguale. In questa maniera possiamo misurare il "volume percepito", che dipende quindi anche dallo spettro di frequenza. I misuratori di decibel sono stati adattati al funzionamento dell'orecchio umano usando

dei filtri che permettono di rispecchiare la risposta dell'orecchio umano. Filtri comuni sono [26] dB(A), dB(B), dB(C).

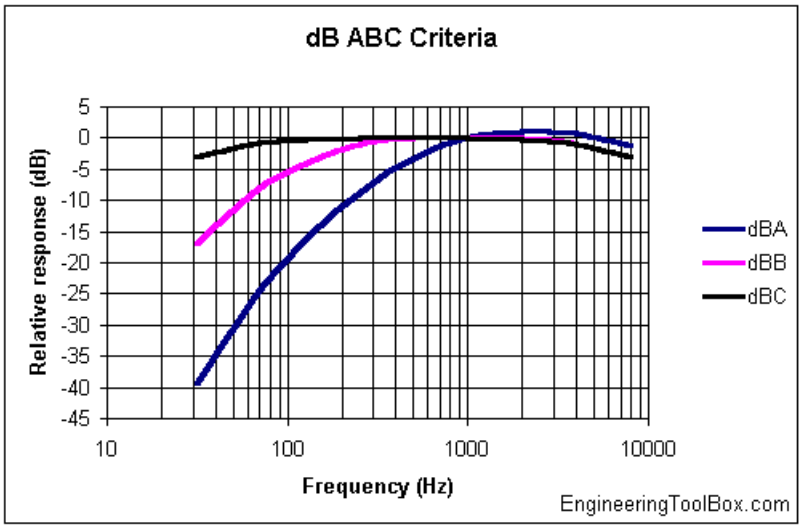

Figura 2.8: filtri dB(A), dB(B) e dB(C) [26]

Approccio psicologico

Questo aspetto è il più difficile da misurare poichè non esistono scale matematiche in grado di fornire una misurazione oggettiva e assoluta. Una conversazione (circa 58 dB) normalmente non viene considerata come fonte di disturbo, tutto cambia però se questa si protae per un lungo intervallo di tempo (anche in orari "legali", ad esempio dalle 09:00 alle 22:00) in quanto renderebbe difficile svolgere molte attività quotidiane (leggere un libro, guardare la televisione, studiare, andare a dormire). Sono numerosi gli esempi che dimostrano quanto la scala dei decibel sia in questo caso inadatta.

Abbiamo visto che il volume di un suono di 60dB è circa il doppio di uno di 55dB. Ciò non significa necessariamente che il suono a 60dB risulti nettamente più fastidioso. Consideriamo i seguenti esempi:

1. Quale suono è più fastidioso?

   (a) 60dB per cinque secondi

(b) 55dB per quattro ore.

2. Qual è il più fastidioso?

   (a) 60dB alle ore 15:00

   (b) 55dB alle ore 03:00.

3. Qual è il più fastidioso?

   (a) una sonata di musica classica a 60 dB

   (b) il rumore a 55dB prodotto da qualcuno che gratta una lavagna.

È evidente che l'approccio psicologico non è da trascurare.

2.2.2 Effetti del rumore sulla natura

Abbiamo visto che il rumore può recare all'uomo danni fisici e psicologici ma cosa succede agli esseri viventi? Spesso si presta poca attenzione ai potenziali effetti che il rumore può avere sulla natura, probabilmente a causa di un comune atteggiamento antropocentrico. È fondamentale osservare gli effetti che il rumore ha sulle altre forme di vita. Siamo disposti a proteggere la fauna dagli attacchi di aerei, dagli elicotteri, dalle navi, da esperimenti scientifici che generano un rumore colossale? Sebbene siano stati effettuati numerosi studi al riguardo non abbiamo ancora prove conclusive sugli effetti negativi del rumore sulla fauna selvatica.

Come abbiamo visto "l'Ipotesi di nicchia" di Krause suggerisce che ogni creatura ha una propria "nicchia sonora". Le vocalizzazioni di tutte le creature di una zona prese insieme forniscono un'impronta vocale che può essere utilizzata per desumere l'integrità biologica della zona. Con la crescente distruzione dei vari habitat naturali molte creature sono costrette in aree diverse, perdendo quindi la loro "nicchia sonora". L'incapacità delle creature di comunicare tra loro presenta dei seri problemi di sopravvivenza delle specie minando quindi l'integrità biologica dell'ambiente.

Determinare gli effetti del rumore sulla fauna selvatica è molto complicato perchè le risposte sono variabili. Questo dipende da vari fattori:

- caratteristiche del rumore e della sua durata;
- caratteristiche proprie di ogni specie;
- tipo di habitat;
- stagione;
- attività al momento dell'esposizione;
- sesso ed età dell'individuo;
- livello di esposizione precedente e se altri stress fisici si sono verificati durante il tempo di esposizione.

Dagli anni 70 ad oggi [22] sono stati condotti numerosi studi per cercare di approfondire la questione. Nel 1975 Dorrance e altri pubblicarono un articolo [14] che sondava il problema degli effetti del rumore del gatto delle nevi sui cervi dalla coda bianca. Tra il 1973 e il 1974 hanno studiato le reazioni di una popolazione di cervi che sono stati esposti al rumore di 195 motoslitte al giorno e le hanno confrontate con quelle di una popolazione che non è mai stata esposta a questo tipo di rumore. Sebbene i cervi esposti al rumore sembravano essersi adattati, si è notato che quelli che non erano mai stati esposti al rumore avevano aumentato il loro *home range* [5] evitando i sentieri della motoslitta. Gli scienziati hanno riconosciuto che questi movimenti extra potrebbero cambiare il bilancio energetico dei cervi, mettendoli quindi in pericolo di salute durante la stagione invernale.

Nel libro "*Noise: The New Menace*" di Lucy Kavaler [16] sono presenti varie sezioni dedicate ai pericoli che il rumore crea alla fauna selvatica. L'effetto più facilmente osservabile è stato un danno all'udito dovuto al danneggiamento delle cellule sensoriali dell'orecchio interno [6] e delle terminazioni nervose adiacenti. Disorientamento e nausa sono state altre risposte comuni.

Presso l'aeroporto *CapeKennedy* di Melbourne (Florida) fu condotto uno studio sul topo del cotone (*Sigmodon hispidus)* che si basava sulla comparazione di due popolazioni

[5] L'*home range* [15] è l'area dove un animale vive. È correlato al concetto di territorio.

[6] l'orecchio interno è la parte dell'orecchio contenente la coclea e i nervi cocleari che trasferiscono gli impulsi elettrici fino al cervello. Si occupa di trasformare lo stimolo meccanico in uno nervoso.

diverse di questo animale: una che viveva presso l'aereoporto, quindi in prossimità di una zona ad elevato rumore e una che viveva a poche centinaia di metri di distanza. La densità della prima popolazione era di 2,58 capi ad ettaro mentre la seconda era di 10,3. Oltre ad una differenza di densità si è registrata anche una differenza comportamentale tra i due gruppi: nel primo erano più timidi e meno propensi alla socializzazione rispetto al secondo gruppo. I ricercatori hanno quindi concluso che queste differenze erano causate dal rumore.

Uno studio sugli effetti dei voli supersonici sulle sterne fuligginose ha mostrato che in Florida fino al 1969 la popolazione media degli uccellini da cova di questa specie è stata di circa 25.000-30.000 capi. Nel 1969 però si è registrato un picco del 99% di morte delle stesse, anno che coincide con il passaggio di questi aerei supersonici sulle zone di nidificazione. Il biologo Dr. W. B. Robertson Jr del National Park Service accusò proprio questi voli militari di essere la causa del disastro, probabilmente perchè il rumore spaventava le madri che per la paura abbandonavano il nido lasciando cadere le uova o abbandonandole semplicemente alla mercé degli altri predatori.

Si è notato che i rumori possono interferire con il sistema di comunicazione degli animali. Il pipistrello ad esempio, affidando la caccia esclusivamente all'eco dei suoni prodotti, può trovare difficoltà nel trovare cibo a causa proprio di queste interferenze. Una minaccia analoga la vivono anche i mammiferi marini e altre specie che utilizzano l'eco per la loro sopravvivenza, per cercare le prede, i compagni o per determinare la rotta delle loro migrazioni. Altri studi sono stati condotti per capire gli effetti degli aerei e degli elicotteri sugli animali. Si è osservato che le reazioni di panico (animali fuori controllo che si urtano, inciampano, etc) e la forte fuga (trotto e corsa per lunghe distanze) sono state le risposte più frequenti quando l'aereo volava al di sotto dei 60 mt. Anche se le reazioni di panico sono quelle più pericolose per la salute degli animali anche le reazioni meno estreme portano alla lontana altrettante conseguenze negative: le lunghe corse nei climi invernali favoriscono malattie polmonari e portano a perdere preziose riserve energetiche, indispensabili per la sopravvivenza degli stessi. Mentre questo studio sembra evidenziare quanto il rumore abbia un impatto detrattivo sulla vita selvaggia altri studi non sono tuttavia d'accordo. Nello studio "La risposta dei tacchini selvatici ai boom sonici" [17] Lynch e Speake piazzarono trasmettitori da 164 megahertz nell'habitat di una ventina di

tacchini selvatici esponendoli a boom sonici. I tacchini generalmente stavano sull'attenti e correvano dai quattro agli otto metri dal punto in cui si trovavano, ma dopo circa trenta secondi ritornavano alle loro precedenti attività. Secondo Lynch e Speake "i boom sonici, come lo studio evidenzia, non inducono i tacchini a risposte che possano metterli in pericolo. La reazione è generalmente lieve e sembrano adattarsi velocemente ai boom sonici". Le differenze tra le conclusioni dei due studi è indicativa delle difficoltà insite in questo settore nella valutazione del problema dell'inquinamento acustico. Una specie può essere più o meno resistente rispetto ad un'altra, rumori diversi hanno effetti altrettanto diversi e anche all'interno della stessa specie ogni individuo può avere risposte differenti a seconda delle differenze fisiologiche o di posizionamento in territori diversi. Conciliare queste difficoltà non è che una delle sfide per gli scienziati e per i politici.

Il rumore prodotto dall'uomo coinvolge anche la fauna marina, spesso infatti il mondo acquatico è sottoposto ad un rumore di intensità anche maggiore. È da poco che si presta attenzione all'inquinamento dell'idrosfera, forse perchè il rumore che non possiamo sentire direttamente ci preoccupa meno. Negli oceani il rumore è creato da numerose fonti tra cui le navi commerciali e militari, dall'esplorazione petrolifera e dagli esperimenti militari e scientifici. Il *"National Marine Fisheries Service"* (che rafforza il *"Marine Mammal Protection Act*" del 1972) nel 1994 sosteneva che spesso gli scienziati nel tentativo di proteggere la vita marina attraverso la loro ricerca "contribuiscono alla persecuzione di questi abitanti del profondo" [18]. In accordo con questa dichiarazione, la società americana ha annunciato in quello stesso anno che il rumore creato dall'uomo costituisce una minaccia sempre maggiore per la salute dei mammiferi marini. La dottoressa Darlene Ketten, specialista dell'udito della *Harvard University,* ha confermato questo sospetto dopo aver trovato frantumate le ossa delle orecchie di due balene. In risposta il *"National Marine Fisheries Service"* propose di porre un limite al rumore marino di 120 decibel, proposta che fu bocciata da molti ricercatori che evidenziarono il fatto che le vocalizzazioni dei delfini si aggiravano intorno ai 130 decibel. Un forte dibattito sull'inquinamento acustico marino è nato in seguito alla creazione del progetto ATOC (*Acoustic Thermometry of Ocean Climate).* Questo programma di trenta milioni di dollari aveva il compito di misurare la temperatura dell'oceano nel tentativo di prevedere i cambiamenti climatici [20]. Utilizzando onde sonore a bassa frequenza e microfoni subacquei nell'Oceano

Pacifico si è riusciti a misurare la temperatura media dell'acqua segnando il tempo di percorrenza del suono prodotto da emettitori sommersi al largo della California e delle Hawaii. Queste onde sonore avevano un volume di 195 decibel e sono state prodotte per sei volte al giorno per dieci anni (il programma è durato dal 1996 al 2006 [21]). Il *National Resource Defence* si espresse in questo modo riguardo la vicenda [19] "Noi semplicemente non possiamo permetterci di giocare alla roulette russa con il nostro sistema globale oceanico".

La questione del rumore nell'oceano non è diversa da quella del rumore sulla terraferma, quanto apprezziamo i diritti degli animali? Quanti sforzi siamo disposti a compiere per garantire quiete agli altri esseri viventi? Le varie divergenze sulla questione e la grande quantità di conoscenze da acquisire sono purtroppo un grave impedimento per comprendere e proteggere la fauna.

Capitolo 3

Soundmap

In questo capitolo viene mostrato che cos'è una soundmap, qual è l'attuale stato dell'arte e quali sono gli obiettivi per la Soundmap da sviluppare.

3.1 Che cos'è una Soundmap

Una soundmap [44], (o mappa sonora) è una forma di media locativo che mette in relazione un luogo e le sue rappresentazioni sonore. La definizione di media locativo indica una comunicazione avente quindi l'obiettivo di rappresentare un luogo evitando la tradizionale distinzione tra codice iconico e sonoro (o verbale). La soundmap veicola contemporaneamente informazioni sull'aspetto visuale e spaziale e dati sull'aspetto acustico e temporale. Le mappe tradizionali tendono a rappresentare in forma bidimensionale una porzione della realtà tridimensionale. Consistono quindi in una rappresentazione semplificata dello spazio con il fine di mostrare la relazione tra gli oggetti che lo compongono. La soundmap invece si pone come strumento senso-motorio, capace di conferire un apprendimento analitico, simbolico, ricostruttivo e un eventuale approfondimento per la riflessione territoriale [44].

« Il nostro senso dell'udito, che finora è stato sottovalutato nella trasmissione e rappresentazione dei dati, può essere utilizzato per ampliare il repertorio rappresentazionale della cartografia. Il suono, in altre parole, ci fornisce ulteriori possibilità per la rappre-

sentazione dei dati e dei fenomeni e più modi per esplorare e comprendere il complesso mondo fisico e umano in cui abitiamo. » (John Krygier, Making Maps with Sound [45]).

3.2 Soundmap analizzate

Prima di procedere alla progettazione e allo sviluppo della soundmap di Trento sono state osservate molte Soundmap esistenti in modo da studiare come vengono attualmente realizzate. Ne riportiamo alcuni esempi per chiarire lo stato dell'arte.

3.2.1 Firenze Soundmap

La soundmap della città di Firenze [46] permette di consultare i suoni della città. La figura 3.1 mostra l'homepage, dopo una breve descrizione in fondo alla pagina è presente la mappa.

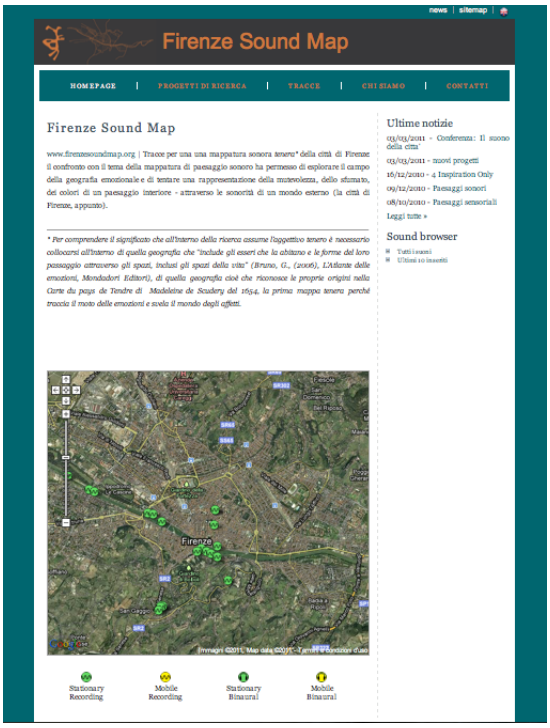

Figura 3.1: Homepage della Soundmap di Firenze

La mappa presenta dei marker che rappresentano i punti dove sono state effettuati i campionamenti. Per ogni punto è stato effettuato un solo campione. Cliccando su un marker appare un popup che mostra il nome del luogo e un player che permette di ascoltare il suono (fig. 3.2).

Figura 3.2: Popup della Soundmap di Firenze

Cliccando sul nome del campione viene caricata una nuova pagina contenente i dettagli (fig. 3.3). Le informazioni mostrate sono relative alla data e agli strumenti di registrazione. È presente a destra la lista dei suoni inseriti che permette di selezionare altri suoni presenti nella mappa. Cliccandoci viene aperto il campione ma non viene mostrata la correlazione con la mappa.

Figura 3.3: Dettagli

3.2.2 New York Soundmap

Similmente alla mappa di Firenze la mappa di New York (fig. 3.4) permette di ascoltare un suono cliccando sui marker presenti nella mappa. L'unico dettaglio sonoro presente è la data. Anche qui è possibile vedere la lista dei suoni inseriti. Si da la all'utente la possibilità di popolare la mappa. È possibile infatti aggiungere dei suoni nuovi cliccando su submit.

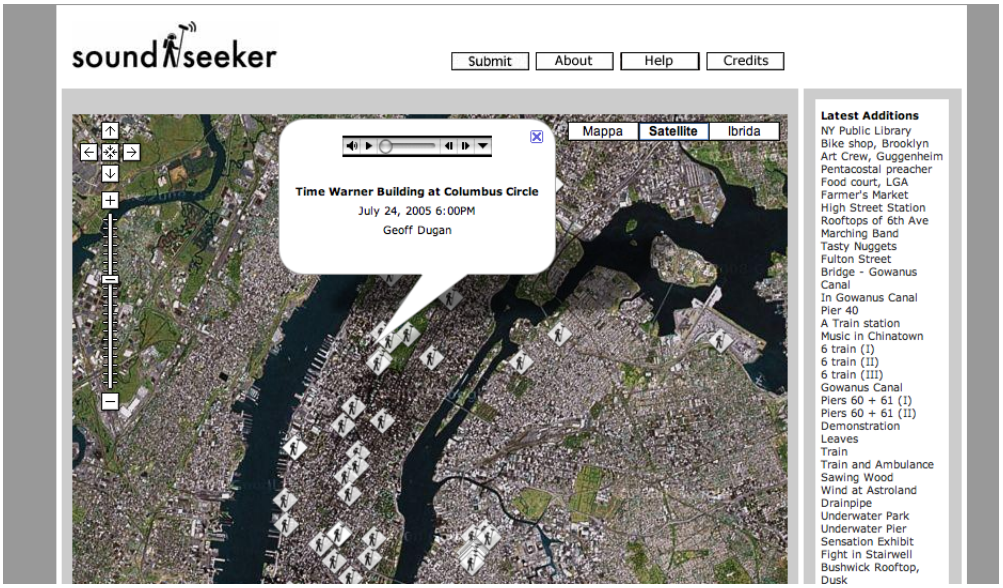

Figura 3.4: Soundmap di New York

3.2.3 Barcellona Soundmap

Appena entrati nel sito bisogna attendere per una decina di secondi che il sistema carichi i suoni. È possibile consultare e scaricare i suoni presenti. Consiste in pratica in una banca di suoni.

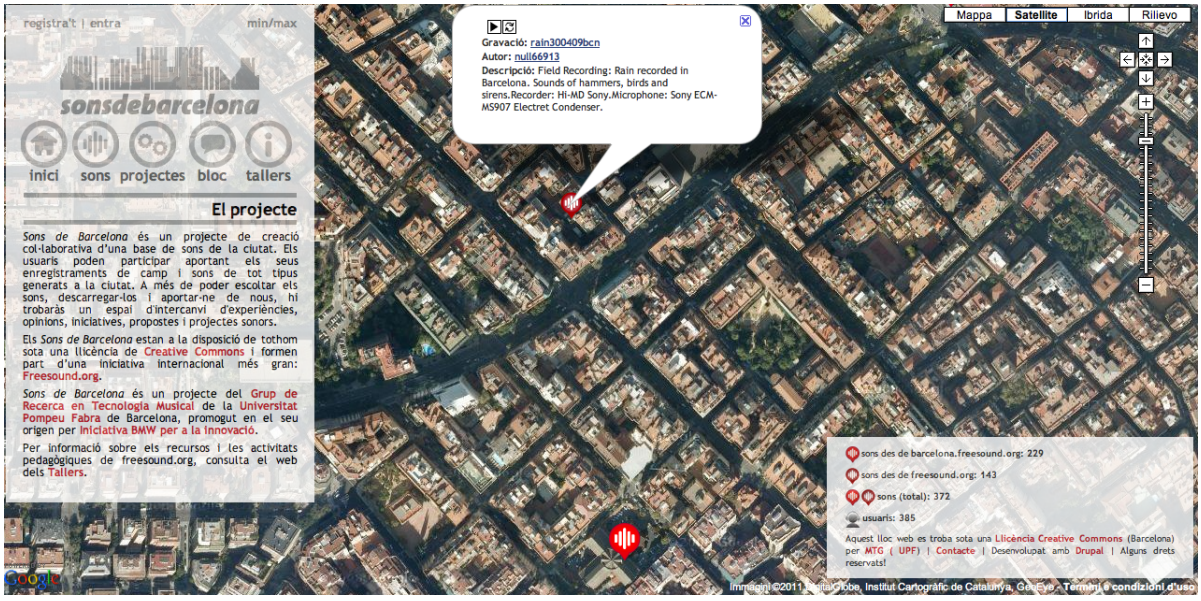

Figura 3.5: Soundmap di Barcellona

3.2.4 New Orleans Soundmap

Questa soundmap è più interessante delle precedenti in quanto permette di effettuare un filtraggio in base a tre categorie del suono. Si può scegliere se mostrare i suoni dell'ambiente, i suoni musicali o le voci. Anche in questa mappa un utente può inserire nuovi suoni.

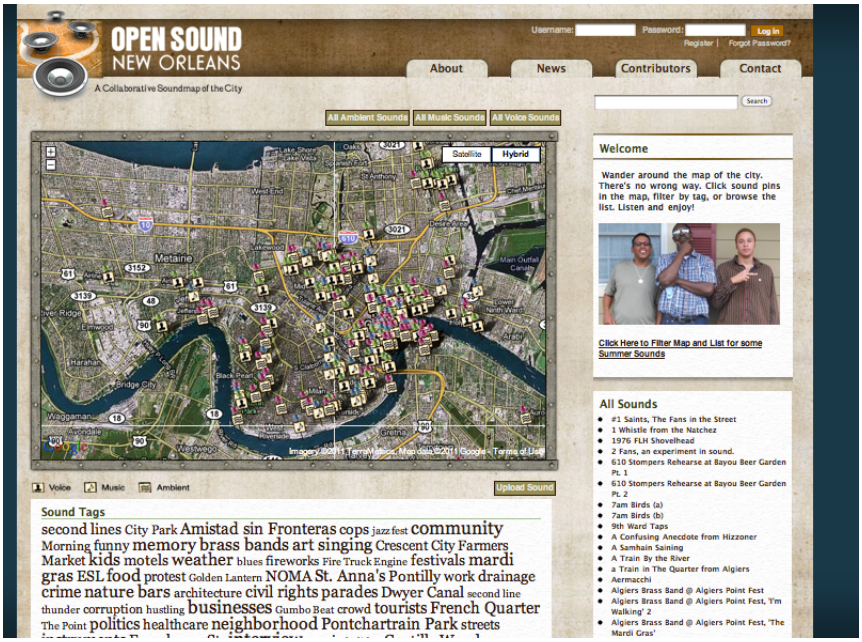

Figura 3.6: New Orleans Soundmap

3.2.5 Uk Soundmap

Le caratteristiche sono sempre le stesse viste finora ma il database è altamente popolato.

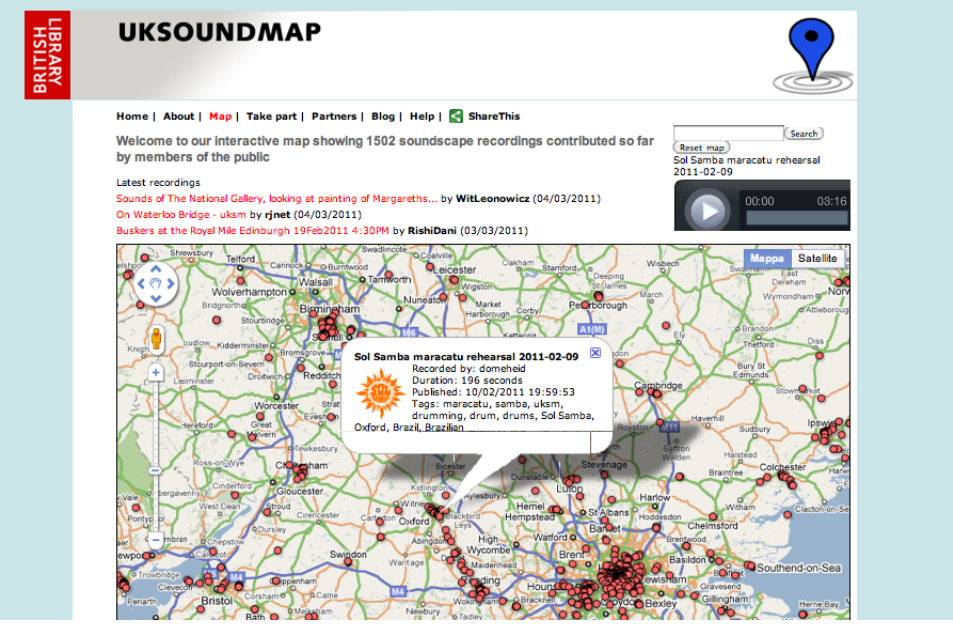

Figura 3.7: UK Soundmap

3.2.6 Montreal Soundmap

Questa soundmap è più interessante perchè mostra le caratteristiche spettrali del campione. Nel browser di destra è possibile cercare i suoni in base a delle proprietà (ad esempio vedere tutti i suoni di Gennaio). Purtroppo i risultati vengono mostrati come una lista di nomi e nella mappa non viene dato nessun feedback, vengono mostrati sempre tutti i suoni.

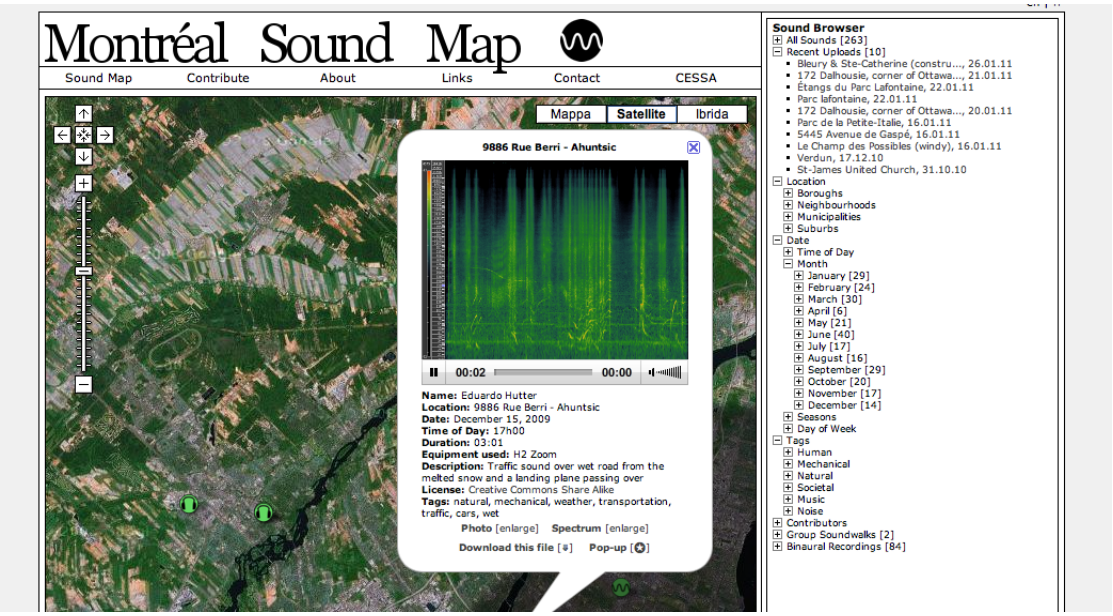

Figura 3.8: Montreal Soundmap

3.3 Obiettivi per la Soundmap di Trento

L'obiettivo di questa tesi è mostrare come la soundmap può essere un potente strumento per monitorare e programmare una tutela del territorio. Rispetto alle altre mappe sonore la soundmap di trento sarà arricchita con ulteriori informazioni di analisi audio.

Ci occuperemo di scegliere dei punti strategici dove effettuare più campionamenti (in giorni e orari stabiliti) in modo da poter osservare "l'andamento sonoro". I suoni verranno quindi analizzati con strumenti esterni. Verrà poi realizzata una soundmap che dovrà essere altamente scalabile e dovrà contenere e mostrare in maniera efficace i dati raccolti.

Come abbiamo visto generalmente le soundmap sono utilizzate per conservare le impronte sonore e "aumentare la coscienza collettiva" su quanto sia importante ascoltare. Il mondo viene inteso come una "composizione musicale" da ascoltare con cura. Per ogni punto viene associato un solo campione e non viene fatta nessun tipo di analisi sui campioni.

Nella soundmap di Trento ogni punto della mappa dovrà contenere più campioni che verranno analizzati in fase di ripresa.

I risultati di questa analisi dovranno essere mostrati dalla soundmap. Per quanto riguarda il singolo campione ad esempio la soundmap dovrà mostrare:

- lo spettogramma (in modo da poter analizzare visivamente lo spettro);
- i valori di picco per alcune bande di frequenza;
- il grado di antropicità del suono (antropico, medio antropico, basso antropico o non antropico);

Verranno misurati i livelli di picco per undici bande di frequenza (50Hz, 100Hz, 200Hz, 500Hz, 750Hz, 1KHz, 2KHz, 5KHz, 10KHz, 15KHz, 20KHz) e saranno messi in correlazione dalla soundmap in maniera completamente automatica. I risultati di questa correlazione saranno mostrati all'utente con dei grafici che permetteranno per ciascun luogo di analizzare come cambia il suono nel tempo.

Per effettuare un efficace monitoraggio dovranno essere presenti dei filtri che permetteranno di esaminare cosa succede in determinati periodi (ad esempio vedere cosa accade

nel weekend alle quattro di pomeriggio). Chiaramente una volta eseguito un filtro la correlazione dovrà essere effettuata unicamente tra i campioni che rispettano i criteri del filtro.

Capitolo 4

Tecnologia

In questo progetto di tesi sono state utilizzate varie tecnologie, tutte basate sul linguaggio Java. Ho utilizzato alcuni componenti della specifica J2EE (in particolare EJB per garantire scalabilità, sicurezza e persistenza dei dati) e li ho integrati con il framework Google Web Toolkit (GWT), utilizzato per la parte presentazionale e per parte della logica. Ho utilizzato l'application server Jboss (l'implementazione certificata JAVA EE 5) sviluppando con l'editor Eclipse. In questo capitolo vengono introdotte le varie tecnologie (in modo da capire le principali caratteristiche), viene mostrato come integrarle tra loro utilizzando l'editor Eclipse e l'application server Jboss.

4.1 J2EE

J2EE è la versione enterprise della piattaforma Java. È costituita da un insieme di specifiche che definiscono le caratteristiche e le interfacce di un insieme di tecnologie pensate per la realizzazione di applicazioni di tipo enterprise e mission critical. Chiunque può realizzare un'implementazione di tali specifiche e produrre application server compatibili con le specifiche Java EE [29].

J2EE è l'acronimo di Java 2 Platform, Enterprise Edition. La parola Java indica il linguaggio di programmazione *cross-platform* sviluppato dalla Sun Microsystem. L'essere *cross-platform* permette ad esempio di sviluppare codice Java su un computer desktop e poi eseguirlo su altri computer, router, telefoni mobili, insomma su un qualsiasi di-

spositivo che sia *Java-enabled*. Questa portabilità è descritta dalla Sun con l'acronimo WORA, che significa *"write once, run anywhere"* (scrivi una volta, esegui ovunque). Un gran numero di mainframes, computer, telefoni mobili e altri device elettronici utilizzano la piattaforma Java. Il 2 nell'acronimo J2EE indica la Versione 2 della piattaforma Java. Dalla versione 5 il numero 2 viene omesso [28] e la piattaforma J2EE viene chiamata Java EE. La versione attuale è la Java EE6. Le due lettere EE sono l'acronimo di *Enterprise Edition*. La Sun ha creato tre edizioni della piattaforma Java:

1. Java ME (Micro Edition), usata nei telefoni mobili e nelle PDAs.
2. Java SE (Standard Edition), che può girare nei device mobili e nei computer laptop e desktop.
3. Java EE (Enterprise Edition), che include tutte le funzionalità delle edizioni Java ME e Java SE e ne aggiunge altre adatte ai server e ai mainframes,

La tecnologia Java EE offre un modello di applicazione distribuita *multi-tiered* [30] che si struttura su quattro tier (fig. 4.1):

1. Client-tier
2. Web-tier
3. Business tier
4. Enterprise information system (EIS)-tier

Di norma le applicazioni *multi-tiered* sono difficili da scrivere perchè devono gestire le transazioni, il *multithreading*, il pooling delle risorse e altri complessi dettagli di basso livello. Le applicazioni Java EE sono invece facili da sviluppare perchè la logica di business è organizzata in componenti riusabili. Questi componenti sono eseguiti all'interno di *container* che offrono numerosi servizi, permettendo allo sviluppatore di concentrarsi unicamente sui problemi di business.

I *container* sono l'interfaccia tra il componente e le funzionalità supportate. Un componente prima di essere eseguito deve essere assemblato in un'applicazione J2EE e inserito nel suo container che fornisce i servizi di middleware necessari al componente. Ci sono vari tipi di container [30] :

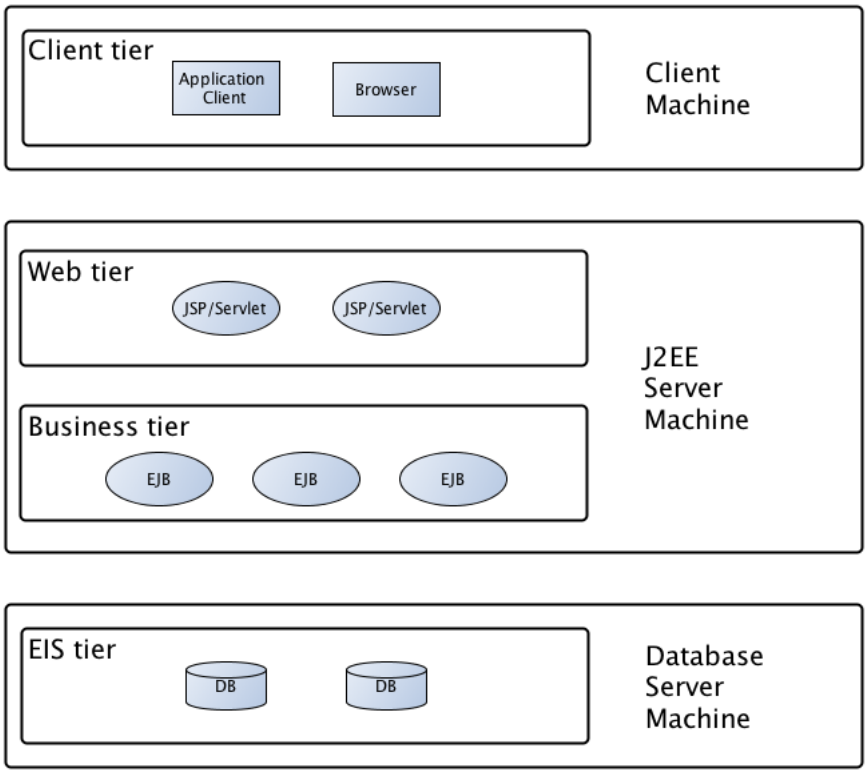

Figura 4.1: Applicazione distribuita *multi-tiered*

- Application client container
- Applet container: gestisce l'esecuzione delle applets. Consiste in un browser Web e un Java Plug-in che girano sul client.
- Web container: gestisce l'esecuzione delle pagine JSP e delle componenti servlet.
- Enterprise Java Beans (EJBs) container: gestisce l'esecuzione degli enterprise beans.

Nella specifica J2EE possiamo distinguere cinque tipi di componenti principali [30]:

- Application clients: girano sulla macchina client. Possono esibire un'interfaccia grafica (GUI) realizzata tipicamente in Swing o in AWT (Abstract Window Toolkit) o avere un'interazione a linea di comando. Possono accedere direttamente agli EJBs eseguiti nel Business Tier. Possono inoltre comunicare con le servlet del Web Tier aprendo una connessione HTTP.
- Web clients: pagine Web (HTML, XML) generate dai componenti Web, browser Web e Applets (piccole applicazioni client scritte in Java ed eseguite nella JVM installata nel Web browser).
- Java Servlets: oggetti Java che dinamicamente processano la richiesta e costruiscono la risposta. Si comportano da livello HTTP-aware e risiedono sul lato server. Comunicano con i client in HTTP e con gli EJBs tramite Remote Method Invocation (RMI). Tipicamente prendono come input una richiesta HTTP, l'analizzano, ci eseguono della logica e poi costruiscono la risposta per il client. Il ciclo di vita delle serlvets è controllato dal Web Container.
- JavaServerPages (JSP): documenti basati su testo. Permettono di creare contenuti Web con componenti statici (HTML, XML..) e dinamici (costruiti da elementi JSP). Sono delle pagine HTML con dei tag JSP che possono contenere codice Java. Il motore JSP traduce la pagine .jsp in codice Java servlet.

- Enterprise Java Beans (EJBs): definiscono un sistema a componenti distribuito che rappresenta il cuore della specifica Java EE. Tale sistema fornisce le tipiche caratteristiche richieste dalle applicazioni enterprise, come scalabilità, sicurezza, persistenza dei dati e altro.

Le applicazioni J2EE sono pacchettizzate in file EAR (Enterprise Archive) che sono dei file JAR (Java Archive) con estensione .ear. I file EAR contengono i moduli J2EE che sono di quattro tipi:

- EJB module: contiene i file .class per degli EJBs e il EJB DD. Il module EJB è pacchettizzato come un file .jar.
- EJB module: contiene i file JSP, i file .class per le servlet, i file HTML e il Web DD. Questo modulo è pacchettizzato come un file JAR con estensione .war.
- Resource adapter module: contiene le interfacce Java, le classi e le librerie e il resource adapter DD. Questo modulo è pacchettizzato come un file JAR con estensione .rar.
- Application client module: contiente i file .class e l'application client DD. È pacchettizzato come un file JAR con estensione .jar.

I *Deployment Descriptors* (DDs) sono documenti XML che descrivono le configurazioni di deployment del componente (es. le informazioni transazionali, il mappaggio sul database etc..). Sono utilizzati per indicare al container come gestire il bean e il suo ciclo di vita.

La piattaforma J2EE offre una serie robusta di servizi middleware [30] come si può vedere in fig. 4.2:

- Enterprize Java Beans (EJBs): definisce come sono scritti i componenti lato server.
- Java Servlets e Java server Pages (JSPs): Definisce e gestisce le servlets e le Jsp.
- Java Remote Method Invocation (RMI) e RMI-IIOP:
  - RMI permette la comunicazione tra i processi.

  - RMI-IIOP è una portabile estensione di RMI che usa il protocollo Internet-Inter-ORB (IIOP) per l'integrazione con CORBA.

- HTTP:

  - HTTP client-side API definite nel package java.net
  - HTTP server-side API definite da interfacce servlet e JSP.
  - HTTPs: HTTP over SSL

- Java Dabase Connectivity (JDBC): è un ponte con i database relazionali che permette allo sviluppatore di invocare comandi SQL da metodi Java. Generalmente è usato dagli EJBs ma a volta anche pagine JSP o Sevlet possono usarlo per accedere direttamente al database. È composto da due parti, un'interfaccia a livello applicazione (usata dalle applicazioni per accedere al database) e un *service provider* per attaccare i JDBC driver alla piattaforma J2EE.

- Java message Service (JMS): permette ai componenti di creare, mandare, ricevere e leggere messaggi.

- Java Naming and Directory Interface (JNDI): offre metodi per associare attributi agli oggetti e cercare gli oggetti usando questi attributi.

- Java Transaction API (JTA): metodi per gestire le transazioni (begins, rollbacks, commits).

- Java mail: usata per mandare notifiche e-mail.

- Java API for XML Processing (JAXP): supporto per processare documenti XML usando il *Document Object Model* (DOM) o le *Simple Api For XML Parsing* (SAX). Permette alle applicazioni di leggere e trasformare documenti XML.

- Java Authentication and Authorization Service (JAAS): permette alle applicazioni J2EE di essere eseguite solo da specifici utenti. Offre quindi meccanismi di autenticazione e autorizzazione.

- Web Service management: offre il supporto sia per i web service client che per i web service endpoints. Include
  - Java API for XML-RPC (JAX-RPC): supporto per le chiamate ai web services usando il protocollo SOAP/HTTP
  - SOAP with Attachments API for Java (SAAJ): offre il supporto per manipolare messaggi SOAP a basso livello.
  - Web Services: definisce il deployment dei web service clients e endpoints. Definisce inoltre l'implementazione dei web service endpoints usando enterprise beans.
  - Java API for XML Registries (JAXR): offre l'accesso al client agli XML registry servers (UDDI)
- J2EE Connector Architecture (JCA): usato per creare resource adapters[1].
- Java IDL: implementazione di un componente base di CORBA basata su Java. Permette l'integrazione con altri linguaggi (J2EE e fully compatible con CORBA).

4.1.1 Enterprise Java Beans 3.0 (EJBs 3.0)

Gli Enterprise JavaBean (EJB) sono i componenti che implementano, lato server, la logica di business all'interno dell'architettura Java EE [31]. Le applicazioni spesso necessitano di risolvere problematiche simili che sono quindi re-implementate dagli sviluppatori, gli Enterprise Java Beans permettono di evitare questo offrendo servizi per gestire in modo rapido e sicuro la persistenza, l'integrità delle transazioni e la sicurezza. Con EJB lo sviluppatore è libero di concentrare le sue energie nel risolvere unicamente problemi di business logic.

Le specifiche EJB descrivono in dettaglio come realizzare un application server che fornisca le seguenti funzionalità [31, 32]:

[1] – componenti software che permettono ai componenti J2EE di accedere e interagire con gli EIS. I resource adapters sono utili per gestire i dettagli di integrazione middleware con sistemi esistenti.

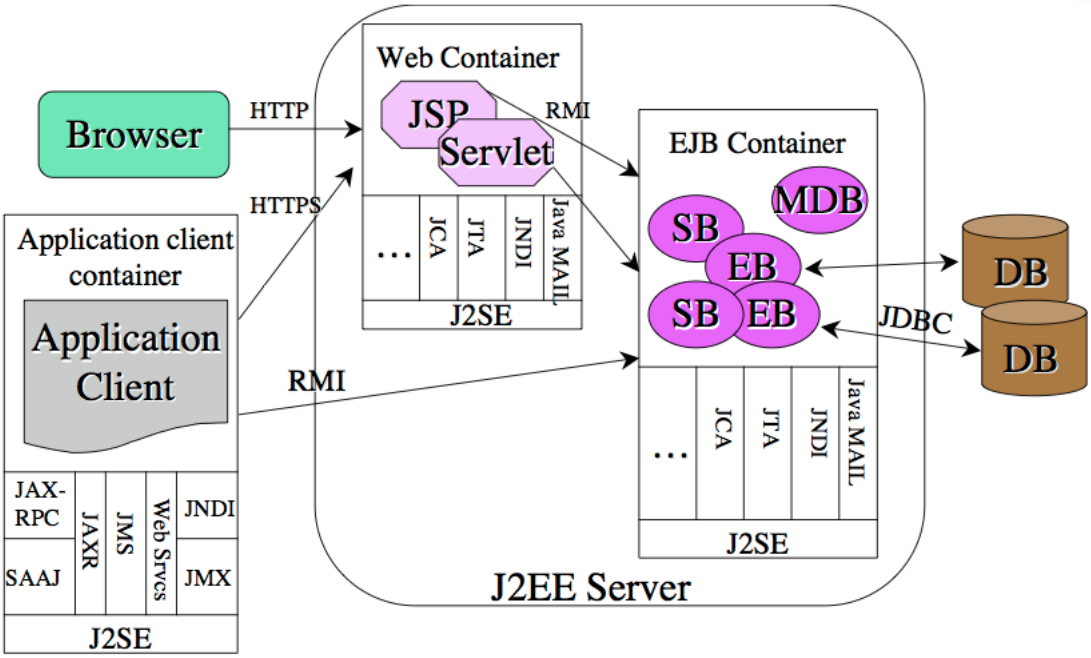

Figura 4.2: Piattaforma J2EE [30]

- persistenza;
- gestione delle transazioni;
- gestione della concorrenza;
- programmazione ad eventi tramite il Java Message Service;
- servizio di directory per elencare e nominare gli EJB (JNDI);
- sicurezza (Java Cryptography Extension (JCE) e JAAS) ;
- installazione di componenti software in un application server;
- invocazione di procedure remote tramite l'utilizzo di RMI-IIOP o CORBA;
- fornire servizi web.

Inoltre le specifiche definiscono il ruolo dell'EJB container e come funziona la comunicazione con gli EJB.

Figura 4.3: Un EJB è un POJO con le annotazioni

Gli EJB [35] sono eseguiti infatti all'interno di un container (che a sua volta è all'interno di un EJB server) che si occupa di fornire numerosi servizi (come ad esempio la gestione delle risorse, il versionamento, la scalabilità, la mobilità, la persistenza, la sicurezza etc..) permettendo allo sviluppatore di concentrarsi unicamente sulle regole di business. Ad esempio se un componente EJB decide di annullare una transazione deve semplicemente comunicarlo al container che si occupa di gestire l'annullamento.

Rispetto alle vecchie specifiche la versione EJB 3.0 ha facilitato notevolmente l'utilizzo degli EJB (semplificando soprattutto la configurazione e l'integrazione con sistemi eterogenei) delegando molti servizi al container. Questo è stato possibile grazie all'adozione del modello di programmazione POJO [2] e utilizzando in modo massiccio le metadata annotation [3] introdotte con Java 5.0. Le annotazioni trasformano un semplice POJO in un EJB (fig. 4.3).

Con queste modifiche l'uso del Deployment Descriptor è diventato opzionale, gran parte del lavoro viene svolto dal container e il modello di programmazione dei bean è molto più semplice. EJB3 ha inoltre semplificato radicalmente il modello di persistenza implementando un approccio Object-Relational Mapping simile a JBoss Hibernate come parte delle Java Persistence Api. La Dependency Injection ha permesso un accesso immediato alle risorse EJB (ad esempio il JDBC DataSource, gli JMS Objects e l'Entity Manager JPA) e ai servizi (come ad esempio Timer, le transazioni utenti e i Web Services) permettendo di dichiarare semplicemente le dipendenze del componente lasciando al container le altre complessità (istanziazione, inizializzazione etc..). In EJB3 la Dependency Injection si può considerare come l'inverso di JNDI, è responsabilità del container

[2] POJO [34] è l'acronimo di Plain Old Java Object. Un POJO è un oggetto Java avente come restrizioni solo quelle delle *Java Language Specification*.

[3] Un'annotazione Java è un modo per aggiungere metadati nel codice sorgente Java che possono essere disponibili per il programmatore durante l'esecuzione usando la Java reflection. Le annotazioni sono compilate all'interno delle classi dal compilatore Java.

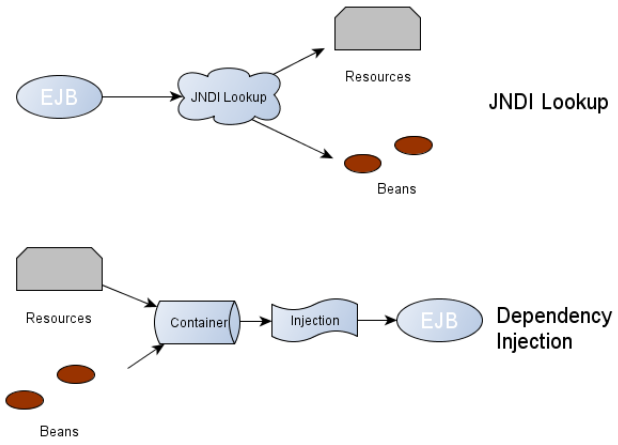

Figura 4.4: JNDI Lookup e Dependency Injection

effettuare l'injection dell'oggetto in base alla dichiarazione delle dipendenze (fig. 4.4).

Esistono tre tipi di EJB [31]:

- Entity Bean: rappresentano i dati persistenti mantenuti nel database. La persistenza è gestita dal container grazie all'utilizzo delle JPA (Java Persistence Api). EJB3 mette inoltre a disposizione un linguaggio proprietario che è possibile utilizzare per interrogare e modificare il database.

- Session Bean: gestiscono l'elaborazione delle informazioni sul server. Generalmente sono un'interfaccia tra i client e i servizi offerti dai componenti disponibili sul server. Il termine Session [33], sessione, identifica il ciclo di vita del componente. Il container utilizza tali oggetti durante una sessione utente, creandoli e distruggendoli dinamicamente. Il loro particolare ciclo di vita non li rende perciò adatti alla persistenza (per la quale esistono gli entity bean). Ne esistono due tipi:

  - Stateful Session Bean, oggetti distribuiti che posseggono uno stato. Lo stato non è persistente, però l'accesso al bean è limitato ad un unico client. L'application server, per mantenere efficienti le risorse, procede a mettere in uno

stato passivo gli stateful session bean che in un dato momento non sono attivi, riattivandoli nel momento in cui l'utente effettua delle operazioni.

- Stateless Session Bean, oggetti distribuiti senza uno stato associato, questa caratteristica permette un accesso concorrente alle funzionalità offerte dal bean. Non è garantito che il contenuto delle variabili di istanza si conservi tra diverse chiamate ai metodi del bean, sono quindi usati quando non c'è necessità di mantenere informazioni tra il client ed il server. Questo particolare modello, che non li lega ad alcun client in maniera forte, rende possibile una gestione di risorse eccellente attraverso il meccanismo del *pooling* [4].

- MESSAGE DRIVEN BEAN: bean con funzionamento asincrono. Tramite il Java Message Service (JMS) si inseriscono in una coda e vengono attivati automaticamente dal container nel momento in cui sulla coda arriva un messaggio. Non richiedono una istanziazione da parte dei client.

4.2 Google Web Toolkit 2.0

Google Web Toolkit (GWT) [36] è un toolkit di sviluppo per creare e ottimizzare applicazioni complesse browser-based. È utilizzato in molti prodotti di Google, come ad esempio Google AdWords e Orkut. È open-source, gratuito e usato da migliaia di sviluppatori in tutto il mondo.

*"GWT's mission is to **radically** improve the web experience for **users** by **enabling** developers to use existing Java tools to build **no-compromise** AJAX for any browser"* [37].

Sebbene GWT sia focalizzato sull'utente (l'obiettivo è creare robuste applicazioni AJAX per migliorare la web experience) presta attenzione anche allo sviluppatore fornendo degli strumenti che rendono possibile la creazione di applicazioni web ad alte perfomance senza dover essere esperti in XMLHttpRequest e Javascript.

[4] L'application server mantiene una struttura dati (pool) in cui esistono una serie di istanze attive di stateless session bean. Ad ogni richiesta viene utilizzato il primo bean del pool libero. Così facendo un pool ben dimensionato può servire migliaia di utenti simultaneamente con pochissimi oggetti disponibili. Riutilizzando quanto già presente in memoria l'application server non spreca risorse nelle operazioni di creazione e distruzione.

4.2.1 Introduzione

GWT gioca un ruolo specifico nella scrittura, nel debug, nell'ottimizzazione e nell'esecuzione del software. (fig. 4.5) :

1. SCRITTURA: quando si scrive codice si ha la necessità di essere molto produttivi. Si necessitano potenti istruzioni, tools aggiornati con lo stato dell'arte, potenti IDE e il refactor deve essere rapido. Per questi motivi per scrivere applicazioni ajax GWT utilizza il linguaggio Java. Le SDK di GWT forniscono numerosi Widget permettendo un rapido sviluppo.

2. DEBUG: utilizzando il linguaggio Java GWT fornisce allo sviluppatore la possibilità di debuggare con facilità (inserire breakpoint, esaminare il contenuto delle variabili). Il Development Mode consente di poter effettuare il debugging della web application in un browser vero e proprio.

3. OTTIMIZZAZIONE: lo sviluppatore non vuole ottimizzare i metodi a mano se il compilatore può farlo per lui. GWT contiene due potenti strumenti per la creazioni di applicazioni web ottimizzate, il compilatore e lo speed tracer. Il compilatore GWT esegue le principali ottimizzazioni del codice (rimozione del dead-code, ottimizzazione delle stringhe etc..). Inserendo i punti di split è possibile inoltre segmentare il download in più frammenti Javascript. "Speed tracer" è un plugin per Chrome che permette di diagnosticare problemi di performance nel browser.

4. ESECUZIONE: per il deployment delle Web Application spesso c'è la necessità di installare plugin aggiuntivi. GWT utilizza unicamente Javascript, HTML e CSS, evitando quindi l'installazione di plugin.

Riassumendo, lo sviluppatore scrive codice Java , GWT lo compila e lo converte in codice Html, Javascript e Css. Questo codice è ottimizzato ed è cross-browser (fig. 4.6).

4.2.2 GWT riduce i tempi di caricamento

GWT riduce drasticamente i tempi di caricamento delle pagine grazie al deferred binding e al raggruppamento delle immagini e al checksum dell'applicazione.

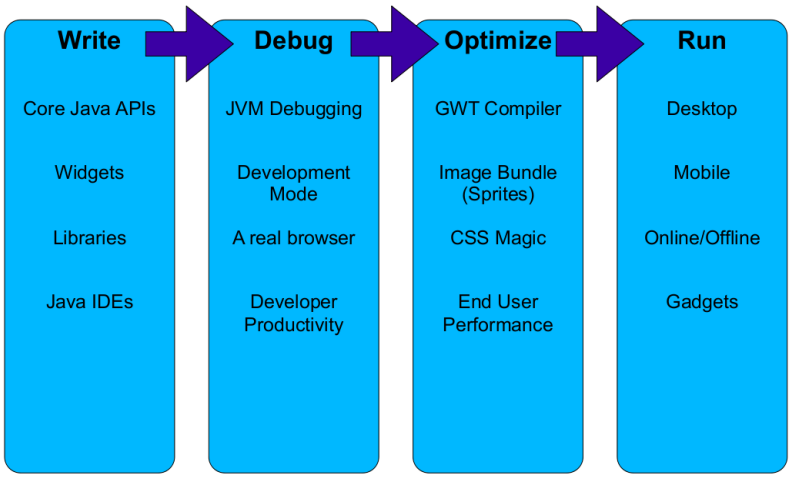

Figura 4.5: GWT nelle quattro fasi [37]

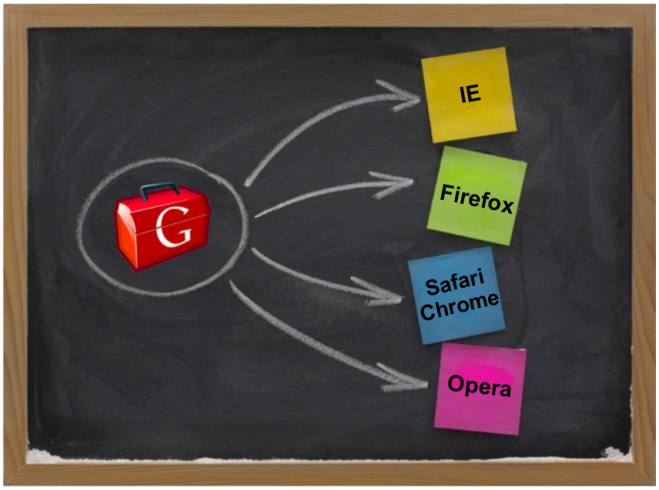

Figura 4.6: GWT genera Javascript funzionante su ogni browser [37]

Deferred binding

Spesso con Ajax si intende modificare dinamicamente parti del testo di una pagina Web. Per fare questo ci sono vari metodi, ed ogni browser per ciascun metodo richiede tempi differenti. La figura mostra i tempi richiesti per i quattro browser più usati ed evidenzia come utilizzare un'unica soluzione supportata da tutti i browser non sia molto performante. Il fatto che spesso la soluzione più efficiente per un browser non è supportata dagli altri porta lo sviluppatore ad un bivio:

1. scegliere il minimo comun denominatore, la soluzione compatibile con tutti i browser a discapito dell'efficienza

2. riempire il codice Javascript di *if-then-else* selezionando a seconda del browser la soluzione più veloce.

La seconda scelta porta ad un unico blocco di codice (contenente le varie versioni) che deve essere scaricato dall'utente. GWT risolve questo problema compilando quattro diverse versioni dell'applicazione. L'utente scaricherà unicamente la versione adatta al proprio browser. Questa soluzione si chiama *deferred-binding* (fig. 4.7).

	Firefox	Webkit (Safari)	Opera	IE
Typical portable `setInnerText()`	2876 ms	1276 ms	2053 ms	4078 ms
`textContent=...`	-	908 ms	1386 ms	-
`innerText=...`	2477 ms	918 ms	1520 ms	2469 ms
DOM manipulation	7148 ms	1997 ms	4836 ms	14800 ms
Improvement	14%	29%	32%	39%

Figura 4.7: Deferred Binding [37]

Image Bundle (immagini raggruppate insieme)

Quando visitiamo un sito ci sono delle risorse da caricare. Tipicamente come prima cosa il browser effettua un dns lookup (ad esempio se visitiamo il sito www.google.com

il browser chiede l'indirizzo ip corrispondente), il dns server risponde e il browser crea una connessione alla porta 80. Questo è un TCP-roundtrip per effettuare la connessione logica tra il web browser (client) e il web server. Il web server risponde dicendo di aver accettato la connessione e il client effettua un terzo passo, stabilisce una connessione http con il server e richiede la pagina html. Il web server restituisce la pagina html che viene scaricata dal client e il browser effettua il parsing e il rendering della pagina. Ogni volta che incontra un tag <img> sa che deve richiedere una nuova risorsa (l'immagine) ed effettua la richiesta al server. Se la web application utilizza ad esempio un css che utilizza molte immagini per ognuna di queste viene effettuata un http-roundtrip. Siccome tra il client e il server c'è una banda limitata una pagina con molte immagini può richiedere tre-quattro secondi per essere caricata. Anche se ci fosse una bandwith infinita la pagina non verrebbe caricata instantaneamente a causa delle latenze. Assumiamo una banda infinita e che il tempo per creare e spedire un pacchetto da qui a New York sia quattro millisecondi per ogni richiesta si perderebbe comunque quel tempo. Uno dei punti chiave per migliorare la User Experience è ridurre il numero di http-roundtrip. Se un sito contiene ad esempio undici immagini GWT le raggruppa in un'unica richiesta riducendo notevolmente il tempo di caricamento 4.8 (la richiesta inoltre è di dimensione inferiore in quanto si elimina l'overhead dovuto ai vari header).

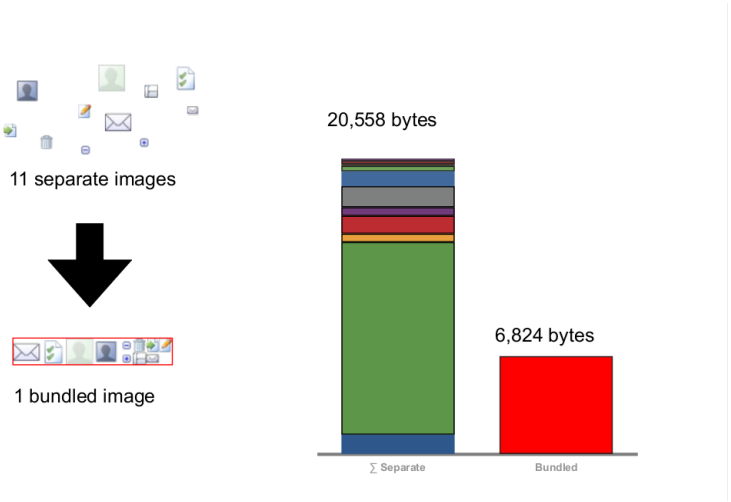

Figura 4.8: Raggruppamento di immagini

Checksum dell'applicazione

Ogni volta che viene rivisitata la pagina viene richiesto al web server se qualcosa è cambiato e il server risponde con la nuova risorsa (se è avvenuto un cambiamento) oppure con un messaggio per indicare che nulla è stato modificato dall'ultima visita. GWT assume che nulla sia cambiato utilizzando un semplice trucco, il checksum dell'applicazione. Quando l'applicazione viene compilata viene calcolato un checksum. Quando qualcosa cambia (ad esempio un'immagine) anche il checksum cambia (si ottiene un nuovo filename). In pratica le risorse vengono segnalate al web server come casuali in modo da non dover controllare se qualcosa è cambiato. Quando c'è un cambiamento l'applicazione viene ricompilata e viene creato un nuovo filename. La vecchia versione dell'applicazione rimane quindi nella cache (evitando di ricaricare alcune risorse come le immagini, il css).

4.2.3 Comunicazione con il server

Tutte le applicazioni GWT eseguono codice Javascript nel browser dell'utente. Spesso però l'applicazione necessita di comunicare con un web server, mandando richieste e ricevendo aggiornamenti. La applicazioni web tradizionali in genere comunicano con il web server e recuperano un'intera pagina HTML. Le applicazioni AJAX invece spostano l'interfaccia logica nel client ed eseguono richieste asincrone al server per mandare e ricevere unicamente i dati. Utilizzando chiamate asincrone l'interfaccia è molto più reattiva. Siccome i motori Javascript nei browser web sono generalmente single-threaded effettuare una chiamata sincrona al server obbliga la pagina web ad attendere il completamento della richiesta. Una rete lenta o un server poco reattivo danneggerebbe l'user experience dell'utente. Con le chiamate asincrone è possibile eseguire altre operazioni in attesa di una risposta dal server. Ad esempio è possibile costruire l'interfaccia utente mentre contemporaneamente si recuperano i dati dal server. È possibile inoltre effettuare più chiamate contemporaneamente[39].

Il meccanismo utilizzato in questa tesi per comunicare con il server EJB è la comunicazione asincrona GWT-RPC [38] che permette alla parte client Ajax di comunicare con la controparte server side che rimane in esecuzione all'interno del server. Dato che l'applicazione client interagisce con l'utente in maniera asincrona rispetto a tutte le even-

tuali chiamate verso lo strato server, il framework mette a disposizione un meccanismo di comunicazione client-server che possiamo considerare come la versione object oriented della comunicazione asincrona alla base di una applicazione Ajax. All'interno di una applicazione GWT infatti è possibile utilizzare un meccanismo di invocazione remota simile ad RMI, ma in cui oltre al modello di invocazione remota a oggetti è presente una efficace e ben progettata semantica asincrona. Il sistema prevede la gestione di notifiche basate su eventi per gestire la risposta asincrona ritornata dal server in modo da aggiornare automaticamente la GUI. In questo progetto di tesi questa comunicazione è stata utilizzata per far comunicare Gwt con EJB.

4.3 Integrazione di GWT con EJB in Eclipse

I componenti utilizzati sono [49]:

1. Google Web Toolkit (version 2.0)
2. Eclipse Galileo, Java EE version
3. GWT Eclipse Plugin
4. JBoss AS 5.1.0.GA
5. JBoss Tools plugin

4.3.1 Creazione dell'ambiente server

Cliccare su Eclipse su *Open Window/Preferences* and selezionare *Server/Runtime Environments*. Cliccare *Add*.

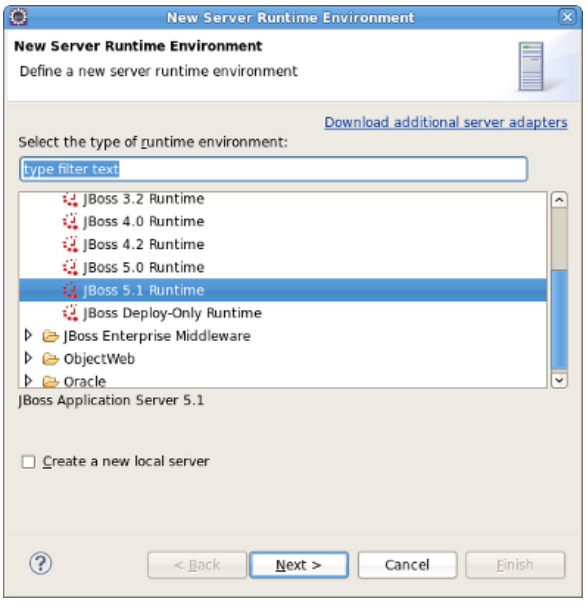

Figura 4.9: Creazione dell'ambiente server

Selezionare *JBoss Community/JBoss 5.1 Runtime*, e cliccare *Next*.

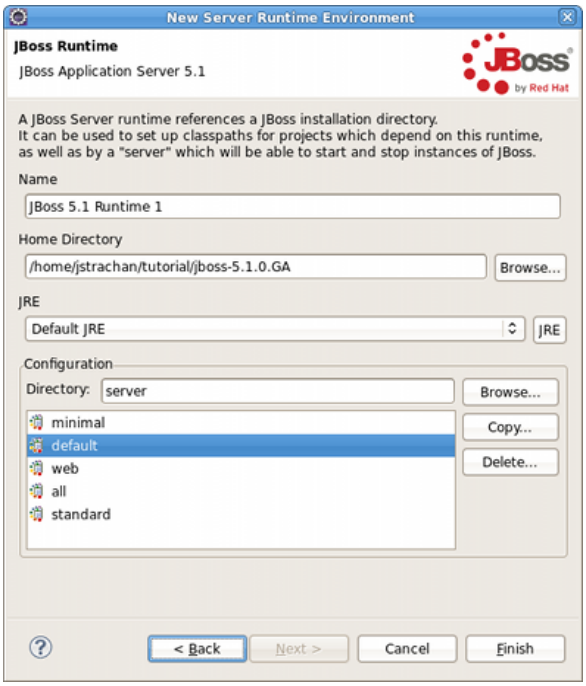

Figura 4.10: Creazione dell'ambiente server

Cliccare *Finish* e poi *OK* per chiudere le preferenze.

4.3.2 Creare un Enterprise Application Project

Selezionare *File/New/Enterprise Application Project* e inserire il nome del progetto

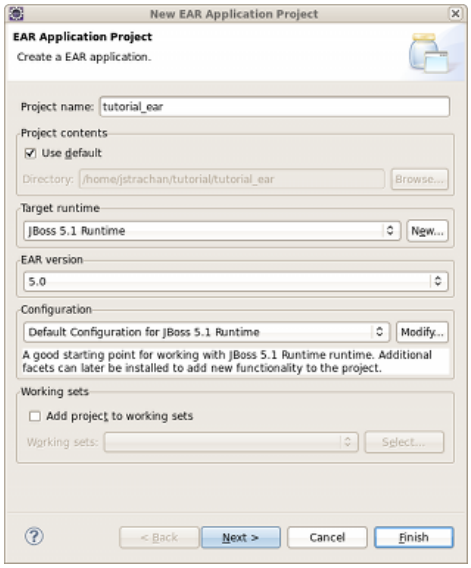

Figura 4.11: Creazione dell'Enterprise Application Project

Cliccare Finish.

Così facendo si è creato un'Enterprise Application aRchive Project che sarà usato per contenere gli altri componenti per un rapido deployment.

4.3.3 Creazione dell'EJB project

Selezionare *File/New/EJB Project* e inserire il nome del progetto. Spuntare Add project to an EAR setting.

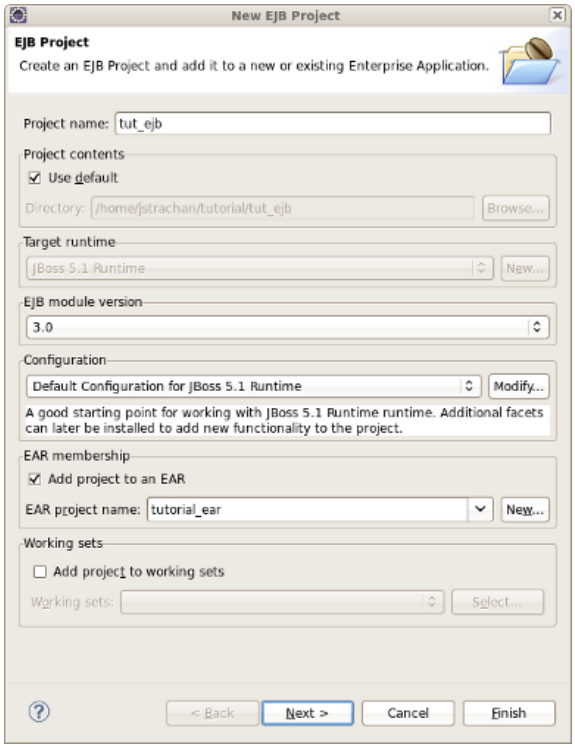

Figura 4.12: Creazione dell'EJB Project

Cliccare Next.

4.3.4 Creazione degli EJB

Creare gli EJB e implementarli con la logica appropriata.

4.3.5 Creazione dei progetti GWT

Per creare le applicazioni GWT basta cliccare sull'icona Google New Web Application Project nella toolbar. Bisogna inserire il nome del progetto, il pacchetto e configurare l'SDK. Deselezionare Use Google App Engine dato che si utilizza JBoss.

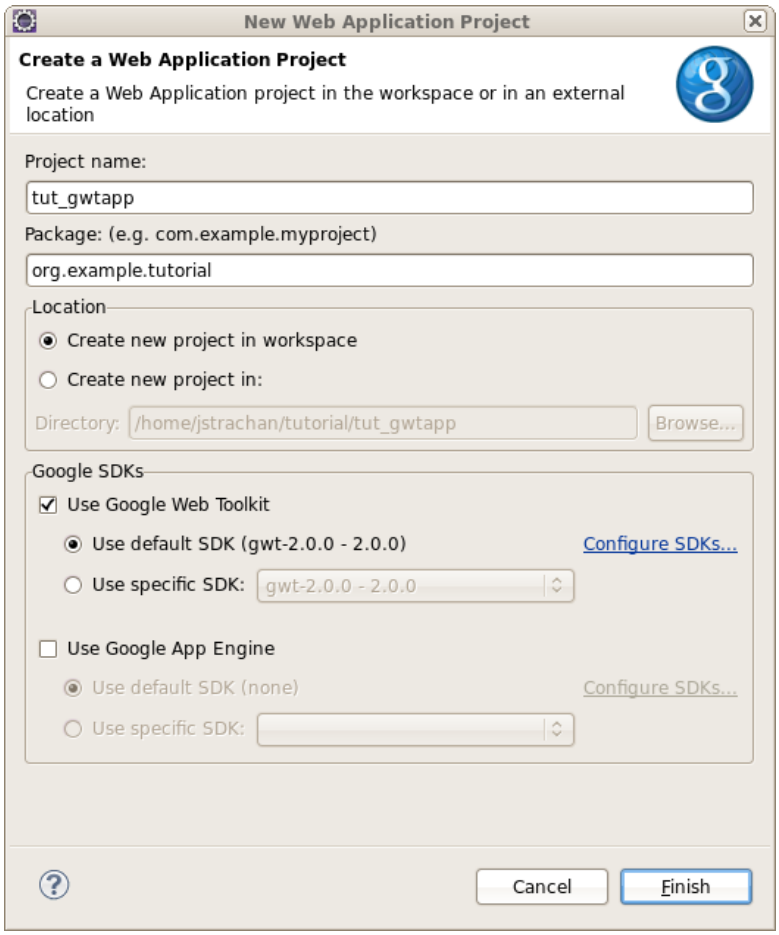

Figura 4.13: Creazione dell'applicazione GWT

4.3.6 Modifica del progetto GWT

Aggiungere facets al progetto

Aprire la *Navigator view* (*Window/Show View/Navigator*), aprire il progetto gwt e cliccare sul file *.project*. Trovare la natures list e aggiungere alcune righe.

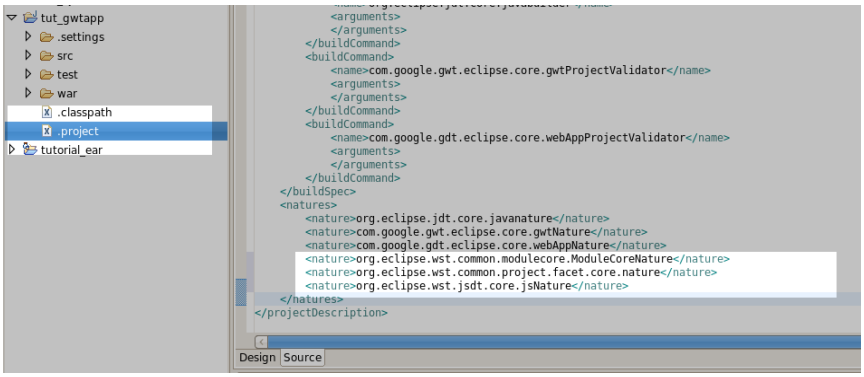

Figura 4.14: facets

Chiudere il progetto e riaprirlo.

Configurare le facets del progetto

- Cliccare con il tasto destro sul progetto e selezionare *Properties.*
- Selezionare *Project Facets*
- Abilitare *Java*
- Abilitare *Dynamic Web Module*

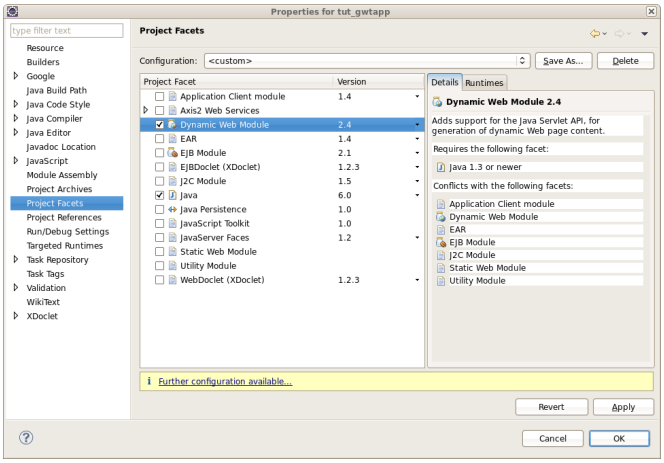

Figura 4.15: configurazione dei facets

- Cliccare su *Further configuration available...*
- Cliccare *Next* e cambiare la directory in ”war”.

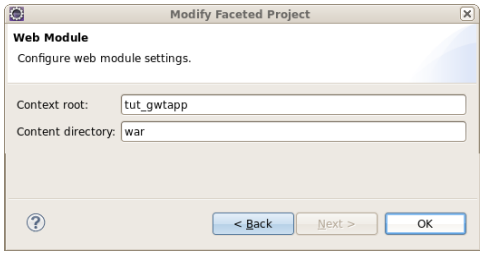

Figura 4.16: Cambio della directory

Modifica del file web.xml

Per far funzionare l’Injection è importante aggiornare il file web.xml dalla versione 2.3 alla versione 2.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/java
ee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd">
```

Figura 4.17: Cambio della directory

4.3.7 Aggiungere il progetto GWT all’EAR

- Cliccare con il tasto destro sul progetto EAR e selezionare *Properties.*
- Selezionare *Java EE Module Dependencies* e selezionare il progetto GWT.

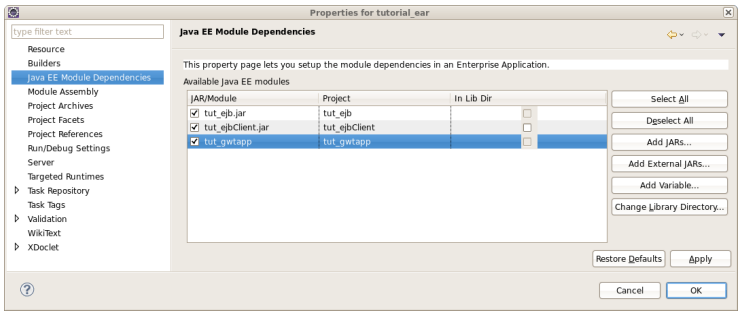

Figura 4.18: Aggiunta del progetto GWT all'EAR

A questo punto l'injection funziona e si può sviluppare l'applicazione.

4.4 Altre librerie

Sono state utilizzate tre librerie fondamentali, Google Maps per la gestione della mappa, il player della bramosystems per la riproduzione audio e Google Image Chart per la creazione dei grafici.

4.4.1 Google Maps

Per la gestione della mappa si è utilizzato la libreria gwt-google-apis [50]. Le API di Google Maps offrono delle Api Javascript che permettono di aggiungere nell'applicazione le funzionalità della mappa. La libreria per GWT permette di accedere a queste API Javascipt dal codice Java compilato con il compilatore GWT. Attualmente le gwt-maps supportano le Google Maps API versione 2.

Per aggiungere la mappa è necessario aggiungere un tag di inherit nel modulo gwt (fig. 4.19).

```
$PP_OFF
    <inherits name='com.google.gwt.maps.GoogleMaps' />
```

Figura 4.19: Inherit

La figura 4.20 mostra un esempio di come aggiungere la mappa nel codice java.

```
public class SimpleMaps implements EntryPoint {

  // GWT module entry point method.
  public void onModuleLoad() {
   /*
    * Asynchronously loads the Maps API.
    *
    * The first parameter should be a valid Maps API Key to deploy this
    * application on a public server, but a blank key will work for an
    * application served from localhost.
   */
   Maps.loadMapsApi("", "2", false, new Runnable() {
      public void run() {
        buildUi();
      }
    });
  }

  private void buildUi() {
    // Open a map centered on Cawker City, KS USA
    LatLng cawkerCity = LatLng.newInstance(39.509, -98.434);

    final MapWidget map = new MapWidget(cawkerCity, 2);
    map.setSize("100%", "100%");
    // Add some controls for the zoom level
    map.addControl(new LargeMapControl());

    // Add a marker
    map.addOverlay(new Marker(cawkerCity));

    // Add an info window to highlight a point of interest
    map.getInfoWindow().open(map.getCenter(),
        new InfoWindowContent("World's Largest Ball of Sisal Twine"));

    final DockLayoutPanel dock = new DockLayoutPanel(Unit.PX);
    dock.addNorth(map, 500);

    // Add the map to the HTML host page
    RootLayoutPanel.get().add(dock);
  }
}
```

Figura 4.20: Aggiunta della mappa

4.4.2 Bramosystems player

Le api bst-player permettono di inserire i plugin dei player più popolari all'interno delle applicazioni GWT.

Gli elementi supportati sono:

- Windows Media Player
- QuickTime player
- Adobe ® Flash® player
- VLC Media Player
- DivX® Web Player plugins
- elementi video HTML 5
- video YouTube

Ogni widget espone metodi per caricare file, iniziare il playback, mettere in pausa, stop, cambiare il volume. Le API includono un plugin detection che permette di mostrare in assenza di plugin un messaggio alternativo (o un widget). In questo progetto si è cercato di utilizzare il player HTML5 (e se il browser non lo supporta il primo supportato).

4.4.3 Google chart API

Le API di Google Chart [47] permettono di generare grafici a partire da una stringa URL. Le API di Google Chart restituiscono un'immagine del grafico come risposta di una richiesta URL GET o POST. Le API possono generare un gran numero di grafici. In questo progetto di tesi sono state utilizzate le *line charts*. Tutte le informazioni del grafico (come ad esempio i dati da passare, le dimensioni, i colori etc..) sono parte della richiesta URL. Questa richiesta inizia sempre con *https://chart.googleapis.com/chart?* seguito dai parametri che specificano i dati del grafico e i dettagli presentazionali. I parametri sono coppie di nome=valore separati dal carattere &. I parametri possono essere in qualsiasi ordine.

Capitolo 5

Architettura

L'architettura della Soundmap sviluppata per la città di Trento consta di tre componenti principali: un progetto EJB per e due applicazioni GWT. La figura 5.1 mostra gli elementi principali del progetto per avere una visione generale dell'architettura (per chiarezza sono stati omessi i dettagli).

Come si può vedere in figura sono state implementate due applicazioni GWT:

1. Soundmap Manager: si occupa dell'inserimento dei dati (campionamenti ed analisi effettuate)

2. Soundmap: si occupa del monitoraggio (rappresentazione dei dati inseriti, possibilità di filtrare i contenuti, correlazione tra le bande di frequenza).

Le parti server delle applicazioni GWT e il modulo EJB sono stati inseriti nell'application server Jboss. L'applicazione Enterprise consta quindi di un pacchetto ear (enterprise archive) nel quale sono stati inseriti il progetto ejb e i due progetti GWT.

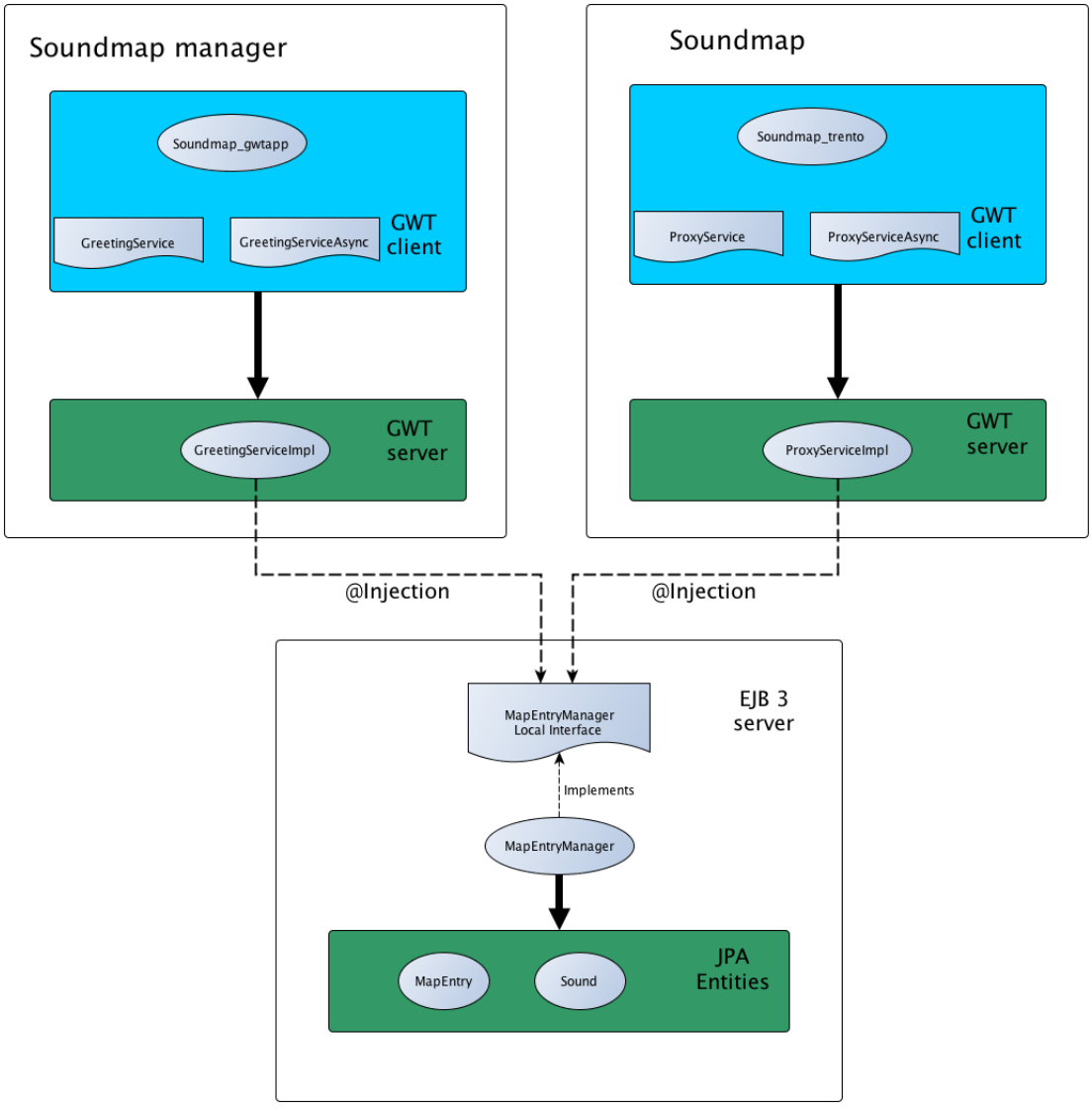

Figura 5.1: Architettura

5.1 Progetto EJB

La Soundmap necessita di conservare un gran numero di informazioni. Per ogni punto di interesse infatti potrebbero essere inseriti numerosi campionamenti, ognuno con molti dati relativi all'analisi del singolo campione. Questo serve sia per avere un campione statistico affidabile (per effettuare analisi accurate) sia per avere una banca dati in modo da preservare le impronte sonore. È necessario quindi avere un'infrastruttura solida in grado garantire sicurezza, affidabilità e scalabilità. Per questo scopo si è utilizzato l'infrastruttura EJB delle specifiche J2EE. La figura 5.2 mostra i pacchetti dell'applicazione sviluppata.

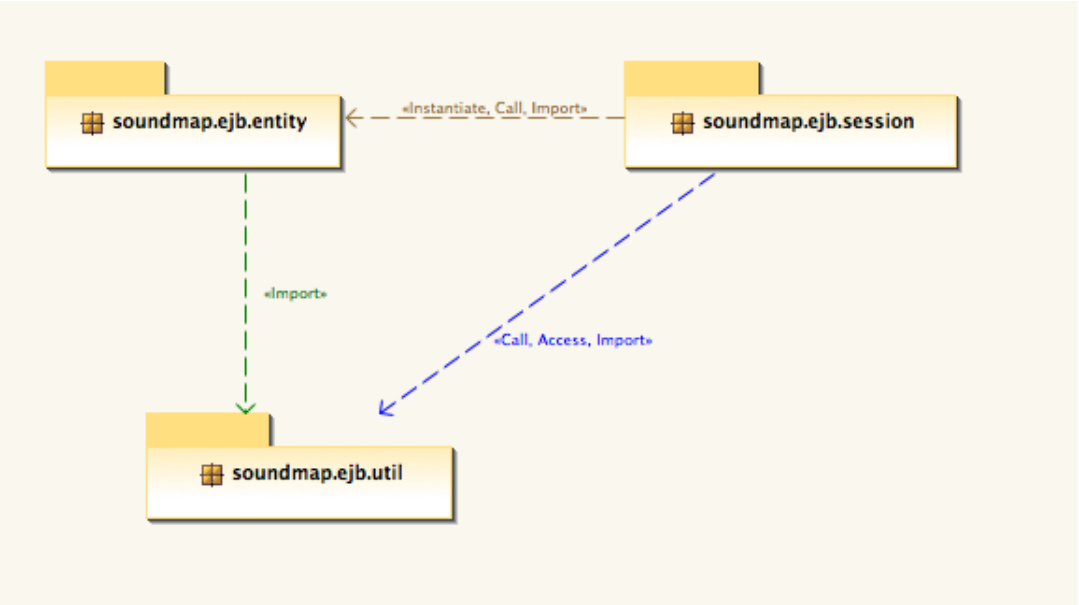

Figura 5.2: Pacchetti dello strato EJB

5.1.1 Entity Bean

Le entità da conservare sono due:

1. MapEntry: punto di interesse dove sono stati effettuati i campionamenti con dati relativi al luogo (coordinate, nome del luogo ed eventuali altri dettagli). L'entity bean corrispondente è mapEntry.java.

2. Sound: singolo campione. È importante conservare oltre all'indirizzo del file audio anche i dettagli estrapolati dall'analisi in modo tale che lo strato di logica permettesse di creare al volo un'analisi incrociata sui dati. L'entity bean corrispondente è sound.java.

In ogni Sound viene mantenuto un riferimento alla mapEntry (fig. 5.3). Questa scelta è stata fatta per permettere di effettuare dei filtri sulla mappa in base alle caratteristiche dei suoni, è possibile infatti visualizzare sulla mappa unicamente i punti dove esistono dei campioni con determinate caratteristiche (ad esempio registrati in un intervallo temporale, oppure con una caratteristica spettrale antropica).

Come si vede in figura 5.3 all'interno del Sound è stato inserito un campo frequency. In questo campo viene mappato per determinate bande di frequenza il relativo valore in dB.

Questo è stato fatto per poter permette di analizzare il comportamento di alcune bande di frequenza tra i vari campioni (ad esempio vedere in un punto specifico l'andamento della banda a 1KHz tra i vari campioni). Sono stati inseriti chiaramente i dettagli relativi alla data di campionamento (in modo da poter cercare i suoni in un certo intervallo di tempo, oppure vedere ad esempio cosa succede solo il venerdì e il sabato) e il grado di antropicità. La classe Coord è embeddable e costistuisce la chiave primaria della MapEntry. Nel Sound invece la chiave primaria è l'url del campione.

Per gestire la persistenza si è utilizzato l'entity manager, un nuovo componente introdotto con EJB 3.0 dedicato alla gestione fisica della connessione con il database da un lato e alla persistenza degli entity dall'altro. Le operazioni di sincronizzazione dei dati fra entities e database sono delegate ad un framework di ORM (Object Relation Mapping) esterno al container. Il fatto di utilizzare un ORM esterno (che in questo caso è Hibernate) ha introdotto la possibilità di lavorare con un livello di dettaglio maggiore

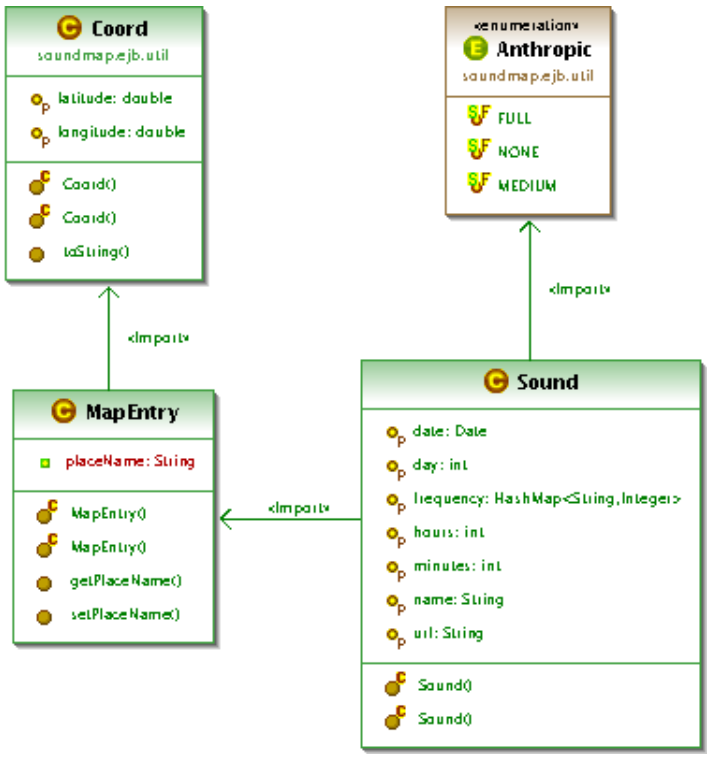

Figura 5.3: Entity Bean

rispetto al passato. Se infatti da un lato si può agire con una astrazione maggiore rispetto a prima (si pensi alle annotazioni che hanno preso il posto dell'XML), sono possibili al contempo operazioni più a basso livello tipiche di strumenti come Hibernate. Ad esempio agendo sull'oggetto EntityManager è possibile eseguire in modo diretto un'operazione di salvataggio per mezzo del metodo persist(). Per quanto concerne invece la ricerca, il metodo find() consente di trovare un entity in maniera diretta utilizzando l'entity bean class name, filtrando la ricerca tramite la chiave primaria. La figura 5.4 mostra un esempio.

```
MapEntry result = (MapEntry) em.find(MapEntry.class, coord);
```

Figura 5.4: Metodo find()

Un Entity Manager [41] deve essere specificato e configurato per mezzo di XML, inserito nel file persistence.xml, contenuto nella directory META-INF. Tale file permette di specificare il nome del database da associare a un particolare schema di persistenza e consente di specificare il comportamento dell'Entity Manager per quanto concerne le regole di traduzione e di mapping dei tipi (dialetto SQL, motore di persistenza da utilizzare etc.). Ogni configurazione contenuta dentro tale file prende il nome di persistence unit; è possibile inserire più persistence unit associate a database differenti, o facenti uso di framework ORM diversi (Hibernate, Toplink etc.). Normalmente i vari parametri di configurazione contenuti in tale file sono specifici dell'application server scelto o del database utilizzato per le operazioni di persistenza. In figura 5.5 è riportato il file persistence.xml utilizzato in questo progetto nell'application server Jboss 5.1.0:

Il persistence unit è stato poi iniettato nel session bean utilizzando la dependency injection 5.6.

Per interrogare il database ed effettuare le ricerche previste si è utilizzato EJB3 QL, un linguaggio molto simile ad SQL con la differenza sostanziale che si referenziano classi ed attributi piuttosto che tabelle e colonne. Si è fatto questo per rendere il codice ignaro del database che si sta utilizzando ed in particolare del suo dialetto SQL. Nell'oggetto EntityManager sono presenti i metodi di generazione delle query. Sono state utilizzate delle query dinamiche parametrizzate (per i filtri più complessi) e delle named query. Le named query sono query precompilate altamente perfomanti che si definiscono all'inter-

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<persistence
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
    version="1.0"
    xmlns="http://java.sun.com/xml/ns/persistence">

    <persistence-unit name="manager">
        <jta-data-source>java:/DefaultDS</jta-data-source>
        <class>soundmap.ejb.entity.MapEntry</class>
        <class>soundmap.ejb.entity.Sound</class>
        <class>soundmap.ejb.util.Coord</class>

        <properties>
            <property name="hibernate.hbm2ddl.auto" value="update" />
            <!--
                <property name="hibernate.hbm2ddl.auto" value="create"
                <property name="hibernate.hbm2ddl.auto" value="create-drop" />
                <property name="hibernate.hbm2ddl.auto" value="update" />
            -->
            <property
                name="hibernate.dialect"
                value="org.hibernate.dialect.HSQLDialect" />
        </properties>
    </persistence-unit>
</persistence>
```

Figura 5.5: File persistence.xml

```
@PersistenceContext(unitName="manager")
EntityManager em;
```

Figura 5.6: Injection del persistence unit

no di un Entity Bean prima della definizione della classe. Sono state utilizzate per le operazioni frequenti (il recupero di tutti i punti della mappa contenenti dei campioni, il recupero di tutti i campioni in un punto specifico). A titolo di esempio una named query è così definita 5.7:

```
@NamedQueries({
    @NamedQuery(
        name="Sound.findSoundByMapEntry",
        query = "SELECT s FROM Sound s WHERE s.mapEntry = :mapEntry ORDER BY s.date"  )
    )
})
```

Figura 5.7: Esempio di named query

5.1.2 Session Bean

Come si vede in figura 5.8 è stato utilizzato un unico session bean stateless per gestire i metodi CRUD [1].

Per le ricerche più comuni sono state utilizzate le performanti named query mentre per le ricerche complesse è stato utilizzato il metodo createQuery dell'entity manager. I filtri sono stati effettuati quindi per mezzo dei metodi findMapEntryByDinamicQuery() e findSoundByDinamicQuery() che permettono di restituire rispettivamente una lista di MapEntry e di Suoni in base alla query inserita in input. Per questo tipo di ricerca non sono state create namedQuery perchè le possibili combinazioni di criteri di filtraggio erano troppi ed è stato quindi più pratico far comporre la query dinamicamente dall'applicazione client.

5.2 Comunicazione di GWT con lo strato EJB

GWT offre un meccanismo di RPC altamente efficiente (integrabile con altri meccanismi RPC come JSON, JSNI o librerie di terze parti) che permette di trasmettere con facilità oggetti Java tramite HTTP. Sfruttando questo meccanismo la logica UI è stata spostata interamente nel client aumentando notevolemente le performance, riducendo la bandwith, diminuendo il carico sul server e offrendo all'utente una fluida user experience. Per semplicità il codice server-side che viene invocato dal client viene considerato come un servizio, quindi l'atto di chiamare una procedura remota viene considerato un richiedere un servizio [40].

[1] Create, Read, Update and Delete (CRUD) sono le quattro operazioni principali per la persistenza [42].

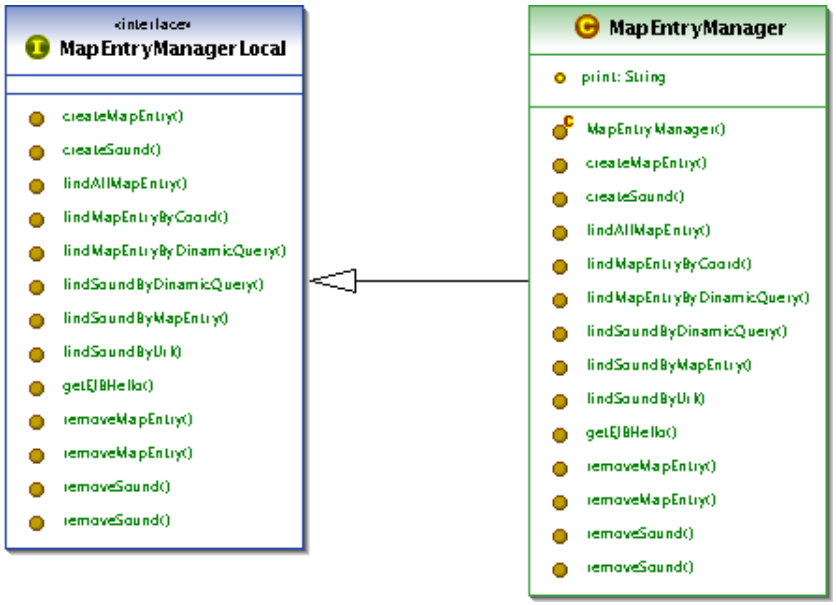

Figura 5.8: Session Bean

5.2.1 Anatomia del servizio RPC

Come si può vedere dalla figura 5.9 entrano in gioco interfacce e classi. Alcune di queste vanno importate (quelle indicate in rosso), altre vanno scritte (quelle blu) ed altre vengono generate automaticamente (il service proxy indicato in verde).

Per definire l'interfaccia RPC è quindi necessario:

1. definire un'interfaccia sincrona per il servizio che estende RemoteService e mostra tutti i metodi RPC;

2. definire una classe che implementi il codice server-side che estende RemoteServiceServlet e implementa l'interfaccia creata.

3. definire un'interfaccia asincrona per il servizio che viene chiamato nel codice client.

5.2.2 Interfaccia Sincrona

Per sviluppare un'interfaccia di servizi è necessario creare un'interfaccia Java che estende l'interfaccia *RemoteService*. Ad esempio [40]:

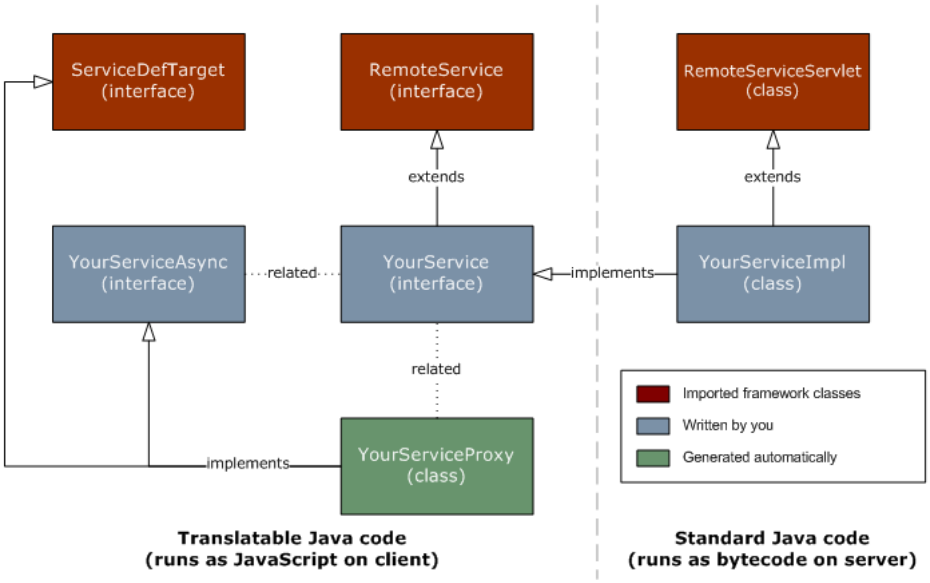

Figura 5.9: Anatomia del servizio RPC [40]

```
package com.example.foo.client;

import com.google.gwt.user.client.rpc.RemoteService;

public interface MyService extends RemoteService {
  public String myMethod(String s);
}
```

Figura 5.10: esempio di interfaccia sincrona

Tutti i servizi richiesti devono quindi essere listati in questa interfaccia. Le implementazioni server-side di questi servizi devono estendere *RemoteServiceServlet* e devono implementare quest'interfaccia di servizi.

È importante tener conto che non è possibile chiamare questa classe direttamente dal client. È necessario creare un'interfaccia asincrona per ognuno dei servizi elencati.

L'interfaccia sincrona nell'applicazione di gestione è *greetingService.java* mentre nell'applicazione di monitoraggio è *proxyService.java.*

Le implementazioni di queste due interfacce consistono semplicemente nell'invocazione dei rispettivi metodi del session bean. Seguendo la configurazione spiegata nell sezione 4.3 è possibile tramite il meccanismo di *Injection* iniettare il session bean (tra-

```
package com.example.foo.server;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;
import com.example.client.MyService;


public class MyServiceImpl extends RemoteServiceServlet implements
    MyService {

  public String myMethod(String s) {
    // Do something interesting with 's' here on the server.
    return s;
  }
}
```

Figura 5.11: implementazione dell' interfaccia sincrona

mite l'annotazione *@EJB*) ed invocare i suoi metodi. La figura 5.12 mostra una piccola parte del file GreetingServiceImpl.java giusto per chiarire il funzionamento.

```
package soundmap.gwt.server;


import java.util.Date;
import java.util.HashMap;
import java.util.List;

import javax.ejb.EJB;

import soundmap.ejb.entity.*;
import soundmap.ejb.util.*;
import soundmap.ejb.session.MapEntryManagerLocal;
import soundmap.gwt.client.GreetingService;
import com.google.gwt.user.server.rpc.RemoteServiceServlet;


/**
 * The server side implementation of the RPC service.
 */
@SuppressWarnings("serial")
public class GreetingServiceImpl extends RemoteServiceServlet implements
GreetingService {


    /**
     * @uml.property  name="me"
     * @uml.associationEnd
     */
    @EJB
    MapEntryManagerLocal me;

    public String greetServer(String input) {
        return me.getEJBHello(input);
    }

    public MapEntry createMapEntry(Coord c, String placeName) {
        return me.createMapEntry(c, placeName);
    }
```

Figura 5.12: Parte dell'implementazione dell' interfaccia sincrona

5.2.3 Interfaccia asincrona

Prima di effettuare una chiamata remota al client è necessario creare un'altra interfaccia client, di tipi asincrono, basata sull'interfaccia originale. Rimanendo in linea con l'esempio:

La natura dei metodi asincroni richiede al chiamante di passare un oggetto callback in modo da poter essere notificato quando una chiamata asincrona viene completata (per

```
package com.example.foo.client;

interface MyServiceAsync {
  public void myMethod(String s, AsyncCallback<String> callback);
}
```

Figura 5.13: Interfaccia asincrona

essere una richiesta asincrona il chiamante non deve bloccarsi in attesa del completamento della richiesta). Per la stessa ragione i metodi asincroni non hanno un tipo di ritorno e ritornano generalmente *void*. Nel caso in cui si voglia avere un maggiore controllo sullo stato di una richiesta in pending è possibile ritornare *Request*. Dopo che viene effettuata la chiamata asincrona tutta la comunicazione col chiamante è passata nel oggetto di callback.

Ci sono degli standard da rispettare per quanto riguarda i nomi delle interfacce. Il compilatore GWT dipende infatti da questi standard per genere il codice per implementare RPC. Un'interfaccia di un servizio deve avere una corrispondente interfaccia asincrona con lo stesso pacchetto e avente come nome quello dell'interfaccia più il suffisso Async. Ad esempio nell'applicazione di monitoraggio la corrispondente interfaccia asincrona per il *proxyService.java* è *proxyServiceAsync.java* mentre nell'applicazione di gestione la corrispondente interfaccia asincrona per *greetingService.java* è *greetingServiceAsync.java*. Ogni metodo dell'interfaccia di servizi sincrona deve avere un metodo corrispondente nell'interfaccia asincrona con un parametro extra di callback passato come ultimo argomento.

5.2.4 Invocare un servizio dal client

Per effettuare una chiamata RPC dal client bisogna seguire i seguenti passi:

1. Istanziare l'interfaccia di servizi usando il metodo GWT.create().
2. Creare un oggeto di callback asincrono per notificare al client quando la chiamata RPC è completata.
3. Effettuare la chiamata.

La fig. 5.14 mostra una tipica chiamata RPC dal client.

```
package com.example.foo.client;

import com.google.gwt.user.client.rpc.RemoteService;

public interface MyService extends RemoteService {
  public String myMethod(String s);
}
```

```
package com.example.foo.client;

interface MyServiceAsync {
  public void myMethod(String s, AsyncCallback<String> callback);
}
```

```
public class Foo implements EntryPoint {
     private MyServiceAsync myService = (MyServiceAsync) GWT.create(MyService.class);

     public void onModuleLoad() {
       // ... other initialization
     }

     /**
      * Make a GWT-RPC call to the server.  The myService class member
      * was initalized when the module started up.
      */
     void doSomething (String message) {
         myService.myMethod(message, new AsyncCallback<String>() {

           public void onFailure(Throwable caught) {
             Window.alert("RPC to sendEmail() failed.");
           }

           public void onSuccess(String result) {
             label.setText(result);
           }
         });
     }
   }
```

Figura 5.14: Esempio di invocazione RPC dal client

5.2.5 Serializzazione

Per effettuare questo tipo di comunicazione è necessario sfruttare il concetto di serializzazione che significa permettere ai contenuti di un oggetto di essere trasmessi e spostati su un'altra applicazione (o salvati fuori dall'applicazione per un'utilizzo successivo). I metodi RPC di GWT e i tipi di ritorno necessitano di essere serializzabili per poter essere trasmessi in rete tra client e server. I tipi serializzabili devono essere conformi a certe restrizioni. Un tipo GWT è serializzabile se

1. il tipo è primitivo (char, byte, short, int, long, boolean, float o double);
2. il tipo è un'istanza di String, Date oppure un wrapper di un tipo primitivo (Character, Byte, Short, Integer, Long, Boolean, Float, o Double);
3. il tipo è un'enumerazione. Le costanti Enumeration sono serializzate unicamente come nomi (i campi interni non vengono quindi serializzati);
4. il tipo è un'array di tipi serializzabili;
5. il tipo è una classe serializzabile.

Questo è un concetto importante perchè ha portato a scegliere di utilizzare unicamente tipi serializzabili. Ad esempio il tipo java.lang.Object (e quindi collection di Object etc..) non è stato utilizzato in quanto non è serializzabile. L'utilizzo dei Java generics [2] ha permesso di rimpiazzare l'uso di istanze di object.

5.3 Progetti GWT

In questo progetto sono state create due applicazioni, una si occupa dell'inserimento dei dati e una del monitoraggio. Per utilizzare il servizio RPC entrambe le applicazioni hanno una parte server e una parte client. Entrambe sono state integrate con le api di Google Maps. La soundmap che si occupa del monitoraggio è stata inoltre integrata con le librerie Bramosystems [48] per il player audio e con le Google Chart Api [47] per la

[2] Le Java gnerics permettono a un tipo o a un metodo di operare con oggetti di vario tipo mantenendo sicurezza dei tipi a tempo di compilazione [43].

creazione dinamica dei grafici di analisi. Per interfacciare GWT con queste librerie è stato necessario modificare il file gwt.xml aggiungendo delle righe di *inherit* (fig. 5.15).

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='soundmap_trento'>
    <!-- Inherit the core Web Toolkit stuff. -->
    <inherits name='com.google.gwt.user.User' />

    <!--  Inherit the Google Maps -->
    <inherits name='com.google.gwt.maps.GoogleMaps' />

    <!--  Inherit the bramosystem player -->
    <inherits name='com.bramosystems.oss.player.core.Core' />

    <!--  Inherit the DateTimeFormat -->

    <!-- Inherit the default GWT style sheet.  You can change       -->
    <!-- the theme of your GWT application by uncommenting          -->
    <!-- any one of the following lines.                            -->
    <inherits name='com.google.gwt.user.theme.standard.Standard' />

    <!-- Other module inherits                                      -->
    <inherits name='soundmap.ejb.entity' />
    <inherits name='soundmap.ejb.util' />


    <!-- Specify the app entry point class.                         -->
    <entry-point class='soundmap.gwt.user.client.Soundmap_trento' />

    <!-- Specify the paths for translatable code                    -->
    <source path='client' />
    <source path='shared' />

</module>
```

Figura 5.15: Soundmap_trento.gwt.xml

5.3.1 Layout

GWT offre numerosi Widget per la creazione di interfacce web particolarmente accattivanti. I programmatori GWT utilizzano spesso strumenti come la FlexTable, un particolare Widget che permette di inserire i contenuti in una tabella , permettendo una rapide creazione del layout. La FlexTable viene però tradotta dal compilatore GWT in una tabella HTML e il suo utilizzo come strumento di layout è quindi da evitare se si vuole separare l'aspetto presentazionale dai contenuti. Per fare questo si è fatto un largo uso del Widget FlowPanel che viene tradotto da Gwt in un HTML div.

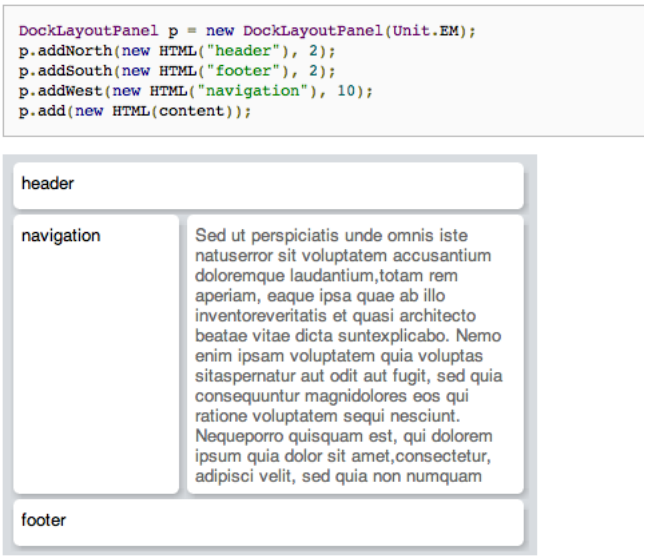

Figura 5.16: Esempio di DockLayoutPanel

Per la struttura della pagina principale si è utilizzato il DockLayoutPanel che struttura la pagina evitando l'uso delle tabelle (fig. 5.16). Utilizzando questi componenti è stato possibile delegare l'aspetto presentazionale ai CSS.

5.3.2 Inserimento dei dati

L'applicazione Soundmap Manager permette di popolare la mappa con nuovi campioni. Cliccando su un punto della mappa è possibile modificare i campioni presenti o aggiungerne dei nuovi. La soundmap di Trento permette infatti di gestire per ogni punto un numero elevato di campioni. Quando si inserisce un suono il sistema chiede varie informazioni del campione (data, ora, grado di antropicità, spettogramma, valore in db per le varie bande di frequenza etc..). Molti di questi dati sono stati estrapolati dopo la fase di campionamento utilizzando software esterni di analisi audio.

5.3.3 Visione dei contenuti

Appena l'utente accede alla soundmap viene mostrata la mappa con sopra dei marker che rappresentano i punti in cui sono stati effettuati i campioni.

La figura 5.17 mostra un esempio.

Figura 5.17: Soundmap di Trento

Cliccando un un marker viene aperto un popup che mostra il campione più recente presente in quel punto. È possibile ascoltare il campione anche in questa visuale (fig. 5.18).

Figura 5.18: Popup

Cliccando su Mostra altre informazioni vengono mostrate le informazioni dettagliate. Viene mostrata la lista di campioni del luogo con i relativi dettagli ricavati dall'analisi.

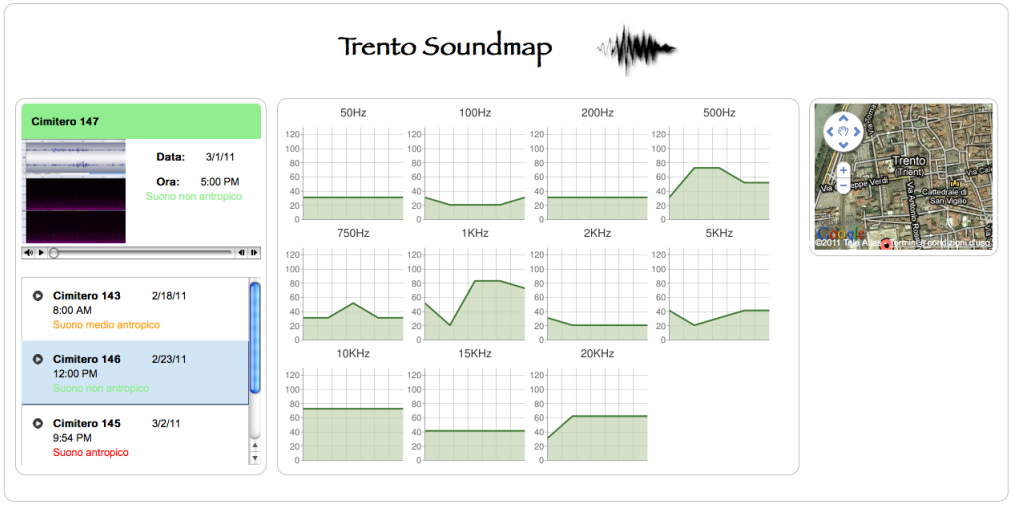

Figura 5.19: Dettagli dei suoni

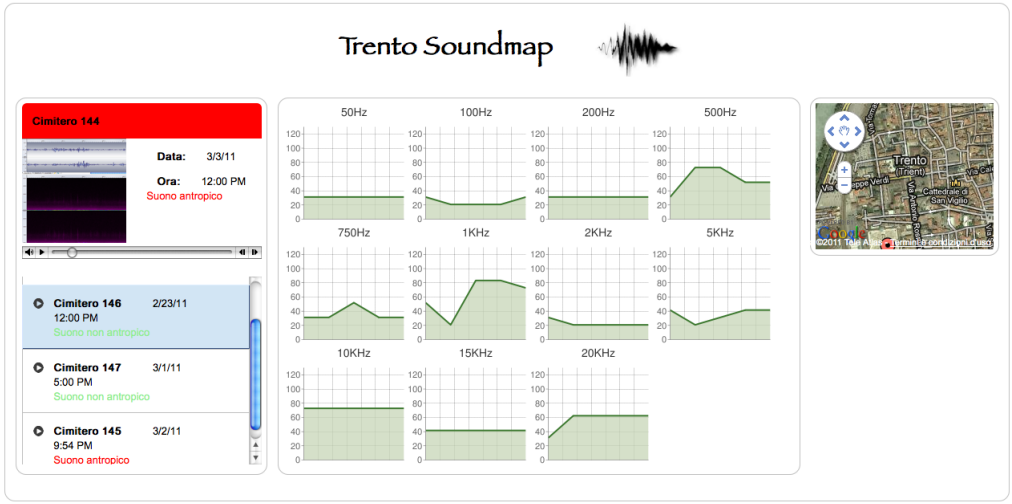

Figura 5.20: Cliccando su un campione della lista si visualizzano di dettagli

Il sistema crea in automatico dei grafici che mostrano l'andamento di alcune bande di frequenza tra i vari campioni. Assumiamo ad esempio di aver preso 50 campioni in un punto specifico. Il sistema mostra come cambia tra i vari campioni il valore in dB della banda a 50Hz, 100Hz, 250Hz, 500Hz, 750Hz, 1KHz, 2KHz, 5KHz, 10KHz, 15KHz e 20KHz.

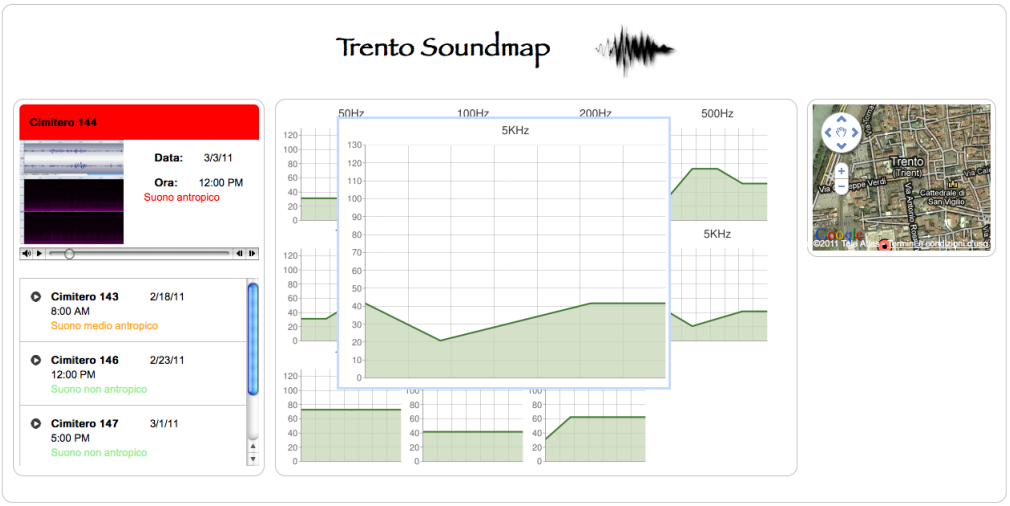

Figura 5.21: Cliccando su un campione della lista si visualizzano di dettagli

5.3.4 Filtri sulla mappa

Per poter effettuare monitoraggio efficiente è spesso necessario analizzare cosa succede in periodi prestabiliti. Il sistema permette di mostrare sulla mappa solo i punti che contengono campioni che rispondono a certi criteri:

1. Intervallo di data;
2. intervallo di orario;
3. giorno della settimana:
4. grado di antropicità

È possibile selezionare uno di questi criteri individualmente o combinarli in qualsiasi modo (ad esempio visualizzare i suoni antropici dei lunedì di gennaio dalle 16 alle 19).

Figura 5.22: Filtri

Quando l'utente clicca su Filtra il sistema risponde visualizzando unicamente i punti dove esistono campioni che soddisfano le proprietà del filtro. Cliccando su un marker e espandendo le informazioni si noterà che per ogni punto saranno presenti unicamente quei campioni e anche i grafici correleranno unicamente i dati di questi. Questa caratteristica è molto utile perchè è possibile analizzare il comportamento di campioni correlati in base a criteri specifici.

Capitolo 6

Analisi e sviluppi futuri

6.1 Strumentazione

Per effettuare i campionamenti abbiamo utilizzato un campionatore Korg MR1 (fig. 6.1).

Le ridotte dimensioni lo rendono perfetto per le registrazioni live. In dotazione viene fornito un microfono esterno stereo a condensatore di ottima qualità, tuttavia è sempre possibile connettere 2 microfoni mediante i 2 ingressi minijack bilanciati su ciasuno dei quali è possibile mediante un commutatore fornire l'alimentazione "Plug in power" (simile alla Phantom ma a basso voltaggio). Tuttavia per non avere alcun tipo di influenza durante i campionamenti abbiamo sostuitito i microfoni a cardioide originali con dei fonometri omnidirezionali con risposta lineare in frequenza.

6.2 Scelta dei luoghi dove raccogliere i campioni

A scopo dimostrativo sono stati scelti alcuni punti "strategici" dove effettuare i campionamenti:

- Corso 3 Novembre
- Cimitero
- MART (museo di arte moderna e contemporanea, vicino alla tangenziale)

Figura 6.1: Campionatore Korg MR1

- Duomo
- Parco del Castello del Buonconsiglio
- Parco in piazza Dante
- Incrocio tra Via San Pietro e Via Manci.

Si è scelto dei giorni e degli orari di campionamento e sono stati registrati i campioni. Questo è stato fatto puramente a scopo dimostrativo, è importante infatti tener presente che questo tipo di campionamento andrebbe effettuato in maniera costante per ogni punto in diversi momenti dell'anno. In un centro urbano per esempio il numero dei campioni dovrebbe essere maggiore perchè le condizioni cambiano sia durante la giornata che durante le stagioni. Su una strada per esempio il suono non è lo stesso tutti i giorni, ma varia anche durante la giornata stessa. Se campiono di notte sarà tipicamente silenziosa, se la campiono all'una alle tre o alle cinque del pomeriggio sara molto più rumorosa. Sui punti urbani bisognerà quindi creare una struttura enorme di campioni in modo da poter monitorare il livello sonico globale presente in quel punto (a livello di inquinamento acustico). Questo tipo di dato ad esempio può essere utilizzato dall'amministrazione

comunale per monitorare se è in qualche modo congruo rispetto alle leggi in ambito di protezione dall'inquinamento acustico. Nelle zone di campagna l'insiediamento urbano è inferiore e chiaramente non serve fare un campionamento ogni ora per tutti i giorni della settimana, si potrebeero diminuire perchè l'attività umana è inferiore. In questo caso si potrebbero incrociare i dati rilevati con quelli relativi all'habitat. Gli habitat infatti sono sensibili non solo all'inquinamento dell'aria, all'inquinamento radiante, a quello dell'acqua e del terreno ma sono sensibili anche all'nquinamento acustico. Le specie viventi presenti all'interno di quel particolare habitat risentono di questo inquinamento subendo inibizioni su una serire di funzioni biologiche:

- comunicazione
- riproduzione
- alcune funzioni cardiovascolari

Questo significa che se il rumore di fondo è particolarmente forte, intenso o particolarmente caratterizzante su alcune frequezne questi animali smettono di comunicare, smettono di riprodurisi, subiscono infarti. Abbandonano l'area.

6.3 Selezione dei campioni

Spesso il campione può avere delle imperfezioni, cioè potrebbe non essere utilizzato. Questo può capitare ad esempio quando nel luogo di campionamento il vento entra nel microfono e crea un bordone sulle frequenze oppure un suono improvviso ma continuo. Tra quelli fatti ad esempio alcuni non erano assolutamente utilizzabili. Poichè statisticamente in alcuni posti ci sono dei campioni che non potrebbero essere utilizzati bisogna sempre prevedere un numero x di campioni in più da inserire. In futuro sarebbe interessante creare un sistema di verifica della congurità del file rispetto al sistema. Come si fa a decidere se un file è attendibile o meno?

Ci sono due modalità:

1. facendo un indagine sullo spettro
2. ascoltandolo

La cosa migliore sarebbe chiaramente ascoltandolo ma sarebbe anche possibile fare delle valutazioni sullo spettro perchè questi tipi di suoni ci sono delle zone con frequenze che sballano tutti i valori. Per il momento abbiamo fatto un'analisi tipicamente uditiva avendo pochi campioni ma dal momento in cui i campioni risulterebbero essere numerosi, sulla base del campionamento di alcuni rumori tipici che possono inquinare lo spettro (rumore del vento, passaggio di un'arereo o di un treno), sarebbe interessante creare una piccola applicazione che mette a confronto questi campioni di rumore col suono campionato per vedere delle similitudini. Si potrebbe sviluppare un'applicazione per cellulari, un'applicazione che permetta alla gente di campionare con facilità e mandare i campionamenti al sistema. Tutti questi suoni prima di essere caricati dovrebbero essere però filtrati. Bisognerebbe quindi creare un sistema di analisi che sia automaticamente in grado di decidere se il suono è coerente o no. Su questo bisognerà lavorare tantissimo perchè non è facile. La tecnologia per farlo però esiste, è la stessa che ha sviluppato Steinberg all'interno dei suoi sequencer e la Celemony con Melodyne, tecnologie di analisi dei campioni dei dati che mostrano determinate cose. Basterebbe avere un set di campioni inquinanti. Se passa l'aereo da queste impronte sonore, il treno ne da altre. Una serie di campioni per cui dire, se nell'analisi dello spettro ti trovi in questa situazione il campione non è valido. Se stiamo ad esempio in alta montagna il passaggio di un aereo creerebbe un boom sonico incredibile. In quel caso se è presente questo boom sonico nel campione il campione risulterebbe non valido.

6.4 Spettrogrammi

Utilizzando WaveLab sono stati analizzati i suoni registrati a Trento e sono stati generati degli spettogrammi.

6.4.1 Spettro

Come si vede nella figura ci sono dei punti gialli che rappresentano i valori di picco. WaveLab7 è un plugin di tipo musicale dove le scale dei decibel non sono asoslute ma partono dallo 0. Lo 0 corrisponde a 100 dB. Quindi -22dB significa 100 - 22 dB cioè 78dB. Come si può vedere in figura 6.2 è presente una barra rossa. Questa barra rappresenta

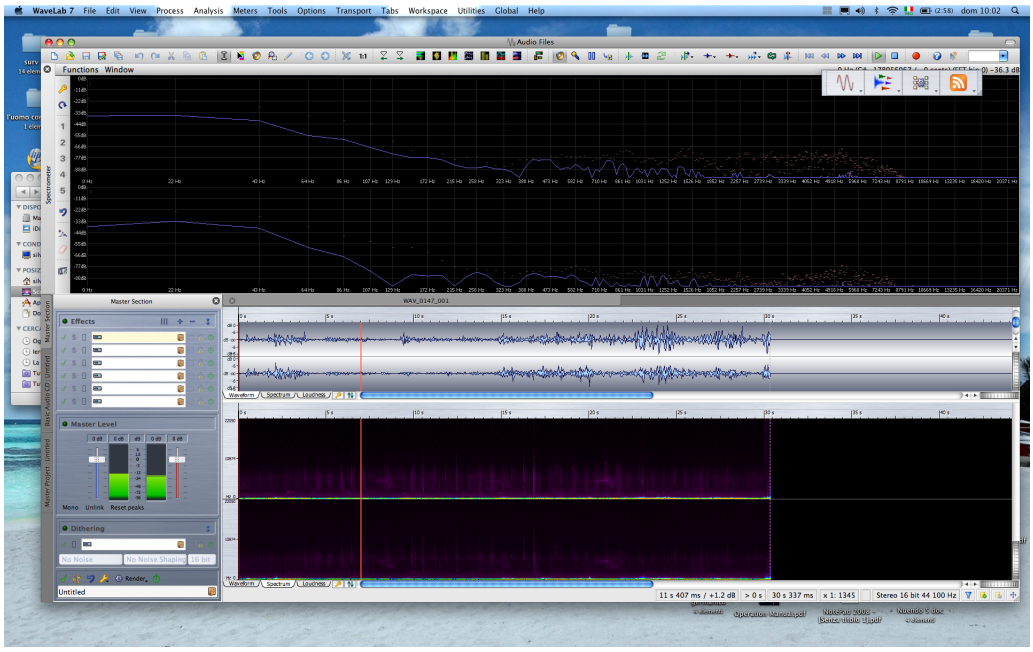

Figura 6.2: Valori di picco

l'instantanea, la curva azzurra si riferisce a quest'istante. Il puntino giallo è stato fatto mandando tutto il file in play, l'applicazione registra i picchi di tutto il file.

Come sviluppo futuro sarebbe interessante inserire all'interno della Soundmap un piccolo plugin che permetta premendo play sul player di vedere in tempo reale questo grafico con le varie frequenze, un analizzatore di spettro in tempo reale. L'utente carica il campione, sente l'audio e vede in tempo reale l'analisi di spettro, la soundmap automaticamente effettuerà l'analisi calcolandi i livelli di picco per le varie bande. L'analisi prevederebbe un'interpolazione in tempo reale per dare un ulteriore valore in più alla qualità dei dati rilevati.

Nella figura 6.3 si notano due spettogrammi, uno superiore uno inferiore. Sono immagini stereofoniche perchè le registrazioni effettuate sono stereo. Da questo spettogramma si piò notare ad esempio che lo spettro non è continuo, ci sono delle linee e c'è una zona dove c'e' una specie di ombra, una specie di nuvola. Queste sono le impronte sonore (in questo caso sono uccellini).

È stato dimostrato che la comunicazione animale avviene per specifiche bande di frequenza, non è diffusa su tutto lo spettro di frequenze. È chiaro che se l'ambiente

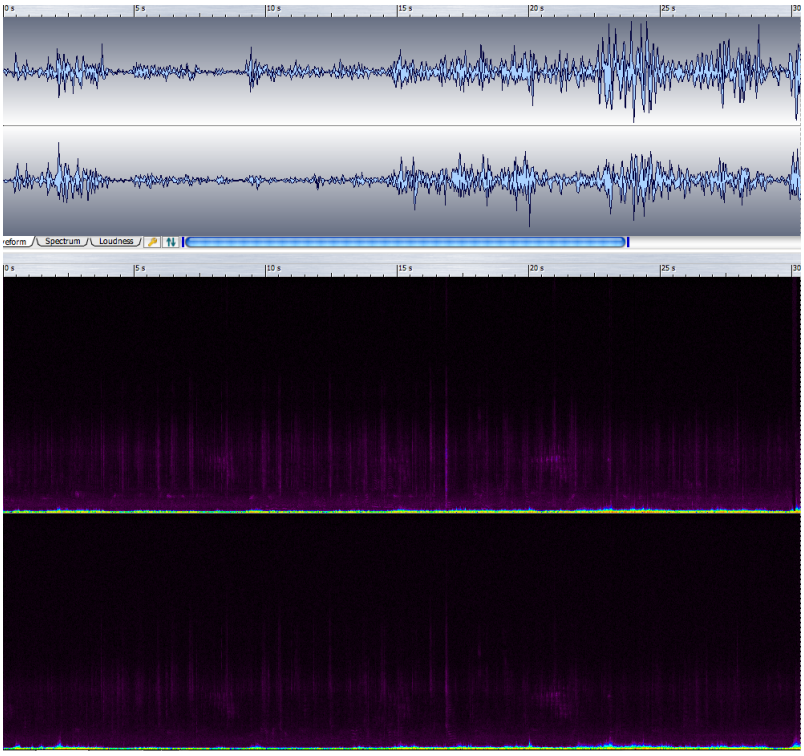

Figura 6.3: Esempio di spettogramma

ha un tipo di habitat particolare dove ci sono delle specie faunistiche che comunicano in quegli spettri di frequenza un eventuale rumore di fondo maschererebbe quella zona di spettro portando alla sparizione di quella specie. Mentre nei centri urbani questi dati vengono utilizzati per il monitoraggio dell'inquinamento ascustico per la salute dei cittadini, nei centri dove l'insediamento umano è ridotto o comunque si integra con gli habitat questi dati servono per monitorare lo stato di salute di questo habitat (se ci sono tanti suoni caratteristici vuol dire che l'ambiente è ricco di specie animali). Purtroppo non è stato possibile incrociare questi dati acustici con dei dati biologi per mancanza di tempo e perchè questi dati in Italia non sono ancora disponibili. Questi dati sono però disponibili presso una serie di società americane che si chiamano Natural Sound Society, società che studiano la corrispondenza che c'è tra la specie animale e il tipo di suono (dicono per esempio: per questa specie animale questo suono è dannoso perchè inibisce queste proprietà). Ciò significa che questo progetto potrebbe avere un ulteriore sviluppo nel settore della classificazione delle specie animali presenti in un determinato habitat in base alle loro caratteristiche di risposta ad uno stimolo acustico. Per poter avere questi dati si potrebbe iniziare una collaborazione con la facoltà di Scineze naturali con un determinato dipartimento si potrebbe migliorare notevolmente la quailtà di un certo habitat riportando per esempio delle spece laddove sono andate via.

Capitolo 7

Conclusioni

Sulla base del lavoro effettuato si evince che integrando le soundmap con alcune applicazioni e plugin si ottiene uno strumento in grado di dare un notevole contributo alla ricerca scientifica. Questo contributo spazia dall'ambito della tutela e della salute dei cittadini a quello della tutela degli habitat in riferimento all'inquinamento acustico. L'applicazione realizzata in questa sede facilita il monitoraggio del territorio permettendo di ascoltare i vari campioni e consultare le informazioni relativi all'analisi (spettrogramma, valori di picco per specifiche bande di frequenza e grado di antropicità). Il sistema effettua automaticamente una correlazione tra i valori di picco di alcune bande di frequenza dei vari campioni e le mostra all'utente con dei grafici (è possibile ad esempio osservare tra i vari campioni l'andamento nel tempo della banda di frequenza a 1KHz) permettendo di studiare come in un punto specifico variano i suoni nel tempo. È possibile inoltre selezionare dei filtri per visualizzare e analizzare solo un sottoinsieme di campioni. Questo permette di osservare i campioni in momenti specifici e correlare i dati in base a dei criteri (si può ad esempio osservare l'andamento nel tempo della banda a 1Khz dei campioni antropici di tutti i weekend dei mesi invernali).

Il monitoraggio potrà essere in futuro ancora più efficiente implementando un plugin in grado di leggere in tempo reale i livelli sonici provenienti dai picchi istantanei dei file. Il database sonoro potrebbe essere popolato dalla gente comune sfruttando un'applicazione per cellulare che permetterebbe a chiunque si trovi in un determinato territorio di effettuare un campionamento ed inviarlo al sistema. Per far ciò attraverso l'analisi delle

frequenze e lo studio di quelle ritenute inquinanti l'applicazione dovrà essere in grado di poter valutare se i file ricevuti dagli utenti accreditati e non siano congrui. Sarà così possibile creare un sistema di analisi e di studio che sia in grado di fare una sorta di GIS sonoro del territorio che i ricercatori di scienze naturali, etologi e biologi potranno ampliare fornendo i dati di risposta delle funzioni vitali delle varie specie animali presenti nel territorio soggette ad impulso sonoro acustico continuato o istantaneo. Questa funzione sarebbe particolarmente interessante in quanto è dimostrato che le funzioni vitali biologiche degli animali sono influenzate dagli impatti acustici, non solo nel sistema della comunicazione ma anche in quello della riproduzione sessuale e del sistema cardio vascolare. Con l'avviamento del progetto di ricerca per il quale c'è l'interessamento del Nuovo Polo Museale di Trento saremo in grado di raccogliere questi dati e di poter creare quindi un sistema che essendo in open source potrà essere esportato e applicato a qualsiasi parte territoriale del pianeta.

Bibliografia

[1] http://en.wikipedia.org/wiki/Acoustic_ecology

[2] Wrightson K., *Soundscape: The Journal of Acoustic Ecology,* Vol. 1, N. 1, 2000, pp. 10-13

[3] http://interact.uoregon.edu/MediaLit/wfae/about/index.html

[4] Monro G., *Soundscape: The Journal of Acoustic Ecology,* Vol. 4, N. 2, 2003, pp. 39-41

[5] Breitsameter S., *Soundscape: The Journal of Acoustic Ecology,*Vol. 4, N. 2, 2003, pp. 24-30

[6] Westerkamp H., *Soundscape: The Journal of Acoustic Ecology,* Vol. 1, N. 1, 2000, pp. 3-4

[7] Westerkamp H., *The Soundscape Newsletter,* N. 1, 1991

[8] http://en.wikipedia.org/wiki/Soundscape

[9] Redstrom J., http://wfae.proscenia.net/library/articles/redstrom_aeecology.pdf

[10] B. Krause, *Anatomy of The Soundscape: Evolving Perspective,* http://www.wildsanctuary.com/popv2/jaes.pdf

[11] R. Schafer, *Il paesaggio sonoro,* Ricordi Lim, Milano, 1985

[12] A. L. Brown, *Soundscape: The Journal of Acoustic Ecology,* Vol. 4, N. 2, 2003, pp. 19-23

[13] V. Reed, *Soundscape: The Journal of Acoustic Ecology,* Vol. 1, N. 2, 2000, pp. 22-23

[14] Dorrance, Michael et al. , *Effects of Snowmobiles on White-tailed Deer*, Journal of Wildlife Management, 1975, pag 563-569.

[15] http://en.wikipedia.org/wiki/Home_range

[16] L. Kavaler, *Noise: The New Menace,* The John Day Company, New York, 1975

[17] Lynch, Speake, *Eastern Wild Turkey Responses Induced by Sonic Booms*

[18] J. Schulhof, *Quiet Please*, Sea Frontiers, 1994

[19] T. Preston, *The Unquiet Oceans*, E Magazine, 1997

[20] Brown, *The Sound of Global Warming*, Popular Science, 1995.

[21] http://en.wikipedia.org/wiki/Acoustic_thermometry

[22] A. Radle, *The Effect Of Noise On Wildlife: A Literature Review*

[23] S. Relandini, *Il suono acustico*

[24] J. B. Tatum, *Physics, Physiology and Psycology,* 1996

[25] G. Ferrington, *Take A Listening Walk and Learn To Listen,* 2007

[26] http://www.engineeringtoolbox.com/decibel-d_59.html

[27] D. White, *What is J2EE, 2011, http*://www.wisegeek.com/what-is-j2ee.htm

[28] Wikipedia*, Java Platform, Enterprise Edition*, http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition

[29] Wikipedia*, Java EE,* http://it.wikipedia.org/wiki/Java_EE

[30] G. Lodi, *Java 2 Enterprise Edition (J2EE),* 2007/2008

[31] Wikipedia, *EJB,* http://it.wikipedia.org/wiki/Enterprise_JavaBeans

[32] Wikipedia, *EJB* ,http://en.wikipedia.org/wiki/Enterprise_JavaBean

[33] Html.it, *Session Bean*, http://java.html.it/guide/lezione/3427/session-bean/

[34] Wikipedia, *POJO*, http://en.wikipedia.org/wiki/Plain_Old_Java_Objects

[35] JavaWorld, *Beans*,http://www.javaworld.com/jw-10-1998/jw-10-beans.html

[36] Google, *GWT,* http://code.google.com/intl/it-IT/webtoolkit/

[37] F. Sauer*, GWT Can Help You Create Amazing Web Apps*, 2009, The San Francisco Java User Group, http://files.meetup.com/930480/SF JUG Oct 13 2009 - GWT can help you create amazing apps.pdf

[38] MokaByte *,La programmazione RIA con GWT/GXT*, http://www2.mokabyte.it/cms/article.run?articleId=IMU-KE6-YFE-AH9_7f000001_18359738_c30510ec

[39] Google*, Communicating with the server*,http://code.google.com/intl/it-IT/webtoolkit/doc/latest/tutorial/clientserver.html

[40] Google, *RPC*, http://code.google.com/intl/it-IT/webtoolkit/doc/2.0/DevGuideServerComm

[41] MokaByte, *Approfondimenti sulla persistenza,* http://www2.mokabyte.it/cms/article.run?art 4DV-BV2-XC8_7f000001_30480431_a64757c3

[42] Wikipedia, *CRUD,* http://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[43] Wikipedia, *Generics,* http://en.wikipedia.org/wiki/Generics_in_Java

[44] Wikipedia, *Soundmap*, http://it.wikipedia.org/wiki/Soundmap

[45] John Krygier, *Making Maps with Sound*, http://makingmaps.net/2008/03/25/making-maps-with-sound

[46] *Firenze Soundmap*, http://www.firenzesoundmap.org/default.asp

[47] Google*, Google Chart Api*, http://code.google.com/intl/it-IT/apis/chart/

[48] *Bramosystems*, http://oss.bramosystems.com/bst-player/bst-player-api/

[49] Integrate gwt with Jboss , http://jamies-gwt.blogspot.com/2010/03/walkthrough-integrating-gwt-with-jboss.html

[50] Google, *gwt maps* ,http://code.google.com/p/gwt-google-apis/wiki/MapsGettingStarted