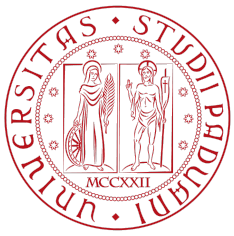


DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

PROGETTO E SVILUPPO DI UN'APPLICAZIONE INTERATTIVA PER IL MUSEO DEI COLLI EUGANEI

Relatore: Prof. Antonio Rodà

Laureando: Massafra Lorenzo

ANNO ACCADEMICO 2023-2024

Data di laurea 21/11/2023

ABSTRACT

Nel campo dell'ingegneria informatica, questa tesi si focalizza sulla progettazione e realizzazione di un web server basato su Processing (versione 4.3). Il progetto è stato commissionato dal Museo dei Colli Euganei di Galzignano Terme (PD), che vuole digitalizzare e rendere più immersiva l'esperienza all'interno della "Sala degli uccelli". Il server, collegato a un web client, interagisce con una web app ospitata su un totem interattivo all'interno del museo. Client e server comunicano su una connessione WebSocket all'indirizzo 'ws://localhost:8025/john'. Quando l'utente interagisce con il totem, il client invia messaggi al server, che risponde mediante la riproduzione dei suoni ambientali e dei canti degli uccelli selezionati. Per ogni regione abitativa viene proiettata la relativa maschera su un plastico 3D del Parco dei Colli Euganei nella sala. Come già menzionato, Processing è stato l'ambiente di sviluppo scelto per questo progetto, è open source e basato su Java, ed è particolarmente adatto alla creazione di effetti visivi, offrendo librerie per l'interfacciamento con dispositivi esterni, quali altoparlanti e proiettori.

I file audio forniti dal museo sono stati opportunamente rinominati per adottare uno standard nomenclativo uniforme, migliorando così l'efficienza del server di risposta ai messaggi. Inoltre, le immagini rappresentanti le zone abitative degli uccelli sono state create ritagliando delle maschere da foto del plastico 3D preesistente nel museo.

Durante la fase di test, sono stati inviati messaggi dal client al server per un periodo prolungato, identificando eventuali anomalie e correggendole nelle prime fasi di sviluppo. I risultati ottenuti sono stati soddisfacenti, evidenziando una notevole responsività e l'assenza di errori da parte del server. Sebbene l'implementazione del programma sia di base e copra le funzionalità minime richieste in fase di specifica, la sua struttura modulare offre opportunità per futuri sviluppi. Questi potrebbero includere l'integrazione di sensori fisici e opzioni di personalizzazione, al fine di arricchire ulteriormente l'esperienza multimediale, specialmente per giovani visitatori provenienti da scuole elementari e medie.

In conclusione, questa tesi rappresenta un significativo progresso nella creazione di un'esperienza interattiva coinvolgente all'interno del Museo dei Colli Euganei.

INDICE

1. INTRODUZIONE

Il museo dei Colli Euganei di Galzignano ci ha incaricato di digitalizzare la fruizione della “sala degli uccelli” che contiene un centinaio di volatili impagliati facenti parte di specie autoctone e abitative dell’area del Parco Nazionale dei Colli Euganei. Il progetto consiste nella realizzazione di un web server basato su Processing, un ambiente di sviluppo open-source ampiamente utilizzato per la creazione di applicazioni interattive e visive. L’idea del progetto nasce dalla registrazione e la riproduzione dei suoni ambientali e del canto degli uccelli in modo da poter ricostruire il soundscape di ben sei habitat che si possono trovare nella zona del Parco. Proprio per questo motivo, prima di addentrarci negli aspetti tecnici della realizzazione del software e dell’interfaccia web server, che permette all’utente di interagire nella sala, ritengo doveroso fare un modesto approfondimento sulla conservazione dei suoni ambientali: su cosa sono, perché sono importanti, in che modo preservarli, quali sono i rischi che minano alla loro conservazione, in quale modo influenzano la percezione dell’ambiente da parte del visitatore e che benefici psicologici hanno sull’ uomo. Inoltre daró una breve panoramica sui recenti studi in ambito eco-sonoro e presenterò diverse realtà internazionali e nazionali che si occupano proprio della preservazione e monitoraggio del soundscape a fini ecologici, urbanistici e culturali.

2. IMPORTANZA DEL SOUNDSCAPE

2.1. I SUONI AMBIENTALI

In primo luogo è opportuno fare una distinzione tra suono ambientale e paesaggio sonoro.
Il paesaggio sonoro (o soundscape) si riferisce all'aspetto più ampio e complesso dell'ambiente acustico. Il termine *soundscape*, coniato dal musicologo canadese Raymond Murray Schäfer nei primi anni Settanta [3a], è, utilizzando la breve definizione di Paul Rodaway [23a], "l'ambiente sonoro che circonda l'ascoltatore", la dimensione sonora dell'ambiente. Nel termine *soundscape* è già inclusa, quindi, una ecologia, una relazione fra soggetto e contesto. Tuttavia, il concetto di paesaggio sonoro può essere descritto come "la colonna sonora della nostra esistenza" [24a]. Questo termine si riferisce all'ambiente acustico percepito o sperimentato da una o più persone in un determinato contesto. Include sia i suoni che percepiamo involontariamente che quelli che cerchiamo e creiamo consapevolmente; tiene conto di tutti i suoni presenti in un dato ambiente, inclusi suoni naturali, suoni umani, suoni artificiali e suoni generati dall'ambiente circostante. In altre parole, il paesaggio sonoro rappresenta un campo di interazioni costituito dalle relazioni tra condizioni geografiche, culturali, sociali ed economiche specifiche di un contesto. I paesaggi sonori del mondo sono in continuo cambiamento, modificati nel tempo in conformità con le mutazioni della società. Il paesaggio sonoro considera come questi suoni interagiscono e contribuiscono all'esperienza sonora di un determinato luogo. È spesso utilizzato in contesti più ampi, come l'architettura del suono e l'ecologia acustica.
Mentre il suono ambientale (o ambient sound) si riferisce specificamente ai suoni naturali o suoni di fondo che si verificano in un ambiente. Questi suoni possono essere rumori come il canto degli uccelli, il fruscio delle foglie, il suono del vento, il gorgoglio di un fiume o qualsiasi altro suono non prodotto direttamente dall'attività umana. Il suono ambientale è una parte del paesaggio sonoro più ampio, ma si concentra principalmente sui suoni naturali che contribuiscono all'atmosfera di un luogo, è un termine utilizzato per descrivere il suono che si verifica naturalmente in un determinato ambiente o luogo. Un'autorevole fonte di riferimento su questo argomento è il libro "The Soundscape: Our Sonic Environment and the Tuning of the World" di R. Murray Schäfer [2a], un pioniere nello studio dell'ecologia acustica.
In breve, il paesaggio sonoro è un termine più ampio e inclusivo che abbraccia tutti i suoni in un dato ambiente, mentre il suono ambientale è un sottoinsieme di questi suoni, specificamente quelli di origine naturale. Entrambi i concetti sono importanti nell'ambito dell'architettura, della progettazione degli spazi e dell'ecologia acustica per comprendere come il suono influenzi l'esperienza umana in diversi contesti [15a].

2.1.2. Classificazione di un paesaggio sonoro

R. M. Schäfer [2a] suddivide le caratteristiche sonore di un paesaggio sonoro nei seguenti elementi:

Toniche: quegli elementi che vengono considerati lo sfondo acustico di un determinato territorio di derivazione naturale come ad esempio le onde del mare, suono degli uccelli, foreste, vento, ecc… . La particolarità delle toniche è che, nel corso del tempo e con la vicinanza mantenuta con questi suoni, è come se essi diventassero invisibili all'orecchio, un'abitudine uditiva alla quale non si fa più caso. Allo stesso tempo, però, trattandosi di suoni fondamentali di un luogo, se dovessero mancare, si sentirebbe la loro assenza.

Segnali: suoni in primo piano ascoltati consapevolmente che svolgono la funzione di avvertimento acustico: clacson, campane, fischi, ecc…

Impronta sonora: suono di riferimento (soundmark) di una determinata comunità. Per la sua unicità contribuisce a determinarne l'identità culturale. Rientra tra le attività del designer acustico la conservazione di tali testimonianze sonore. Le impronte sonore, pertanto, devono essere preservate e riconosciute come elemento d'identità da salvaguardare e proteggere proprio perché danno valore al luogo stesso in cui si trovano.

Tali elementi possono talvolta risultare ambigui e trovare una collocazione diversa in paesaggi differenti [14a]. R. M. Schäfer [2a] classifica i paesaggi sonori anche in base alla possibilità di ascoltare e riconoscere suoni distinti:

Paesaggio sonoro Hi-Fi: con un basso rumore di fondo ambientale. Consente di percepire distintamente i suoni (es. un bosco).

Paesaggio sonoro Lo-Fi: con un alto rumore di fondo ambientale. Non consente di percepire distintamente i suoni né la loro provenienza (es. traffico in città).

Bernie Krause [4a] ridefinisce in tre categorie gli elementi del paesaggio sonoro tenendo conto dell'origine del suono:

Geofonia: suoni prodotti da quattro elementi naturali: vento, acqua, terra, fuoco e tutte le combinazioni tra loro.

Biofonia: suoni degli organismi viventi, uomo escluso.

Andropofonia: suoni creati dall'uomo. Musica, linguaggi, suoni creati da macchine elettromeccaniche, ecc…

2.1.2. Eco-acustica

L'eco-acustica rappresenta in qualche modo una filiazione nel campo di studi della bio-acustica e si occupa più specificamente di indagare le relazioni presenti all'interno degli ecosistemi, da un punto di vista ecologico [21a] [18b].
Il compositore eco-acustico è colui che, ponendosi all'interno di questo ambiente, che sta sempre più diventando un settore di ricerca scientifica, tenta di capire cosa succede nella complessità degli habitat naturali primari e lavora soprattutto con le relazioni presenti all'interno di tali ecosistemi.
La composizione eco-acustica rappresenta quindi quell'agire creativo contraddistinto, però, da un forte imprinting scientifico di studio analitico sugli ecosistemi, secondo la definizione di David Monacchi. [22a]
Schäfer nel "Paesaggio sonoro" (1985) [1a] sviluppa poi il concetto di "ecologia acustica", intesa come "lo studio dei suoni nel loro rapporto con la vita e la società". L'autore scrive che essa può essere indagata solo sul campo, andando ad analizzare l'influenza dell'ambiente acustico sulle persone che vi abitano. L'opera si concentra sull'analisi e la comprensione dei suoni presenti nell'ambiente circostante, sia naturale che antropico. Schäfer esplora come il suono influenzi la nostra percezione dell'ambiente e come sia importante nella formazione delle nostre esperienze quotidiane. Il libro analizza come i suoni contribuiscano all'identità e all'esperienza dei luoghi. Inoltre, Schäfer discute di come i cambiamenti nel paesaggio sonoro possano avere un impatto sulla qualità della vita e sulla salute umana. Nel complesso, l'opera promuove una maggiore consapevolezza dell'importanza del suono nell'ambiente e nella cultura umana. A tal proposito, Andrea Minidio [6a], nel suo testo "I suoni del mondo", riporta l'esempio della tribù Maaban, che occupava una zona del Sudan priva di rumori legati alla società meccanizzata e che, pertanto, presentava un udito migliore di quello delle civiltà industrializzate. L'ecologia acustica, però, si occupa anche delle relazioni che intercorrono tra le espressioni musicali di un popolo e l'ambiente in cui quello stesso popolo abita. Pertanto, questa disciplina si serve non di rado degli apporti e degli studi elaborati all'interno della geografia.

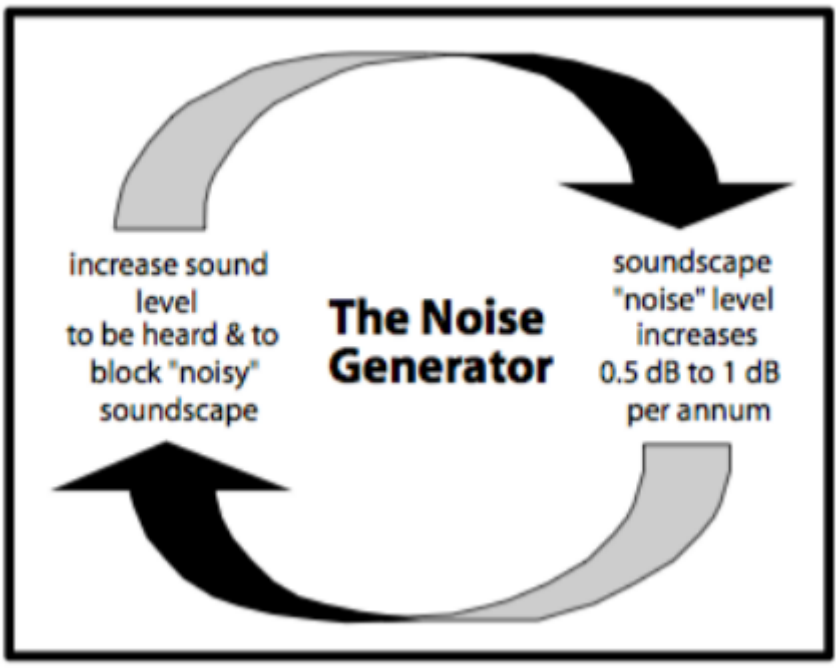
2.1.3 I suoni della città

Il paesaggio sonoro urbano è un aspetto fondamentale dell'esperienza quotidiana delle città. Michael Bull, in "The Auditory Culture Reader" [7a], afferma che "la città moderna, per quanto ci sia molto da vedere, non è solo spettacolare: è sonora". Questa affermazione evidenzia l'importanza dei suoni urbani nella creazione dell'identità della città. La città è caratterizzata da una varietà di suoni che contribuiscono a definire il suo paesaggio sonoro. Bull sottolinea come i suoni del traffico siano diventati le "sigle musicali delle città moderne", indicando che i suoni della città sono spesso legati al movimento. Questi suoni, come il rumore del traffico o il passaggio di treni, diventano parte integrante del tessuto urbano e influenzano la percezione e l'esperienza delle persone.
Spesso però la città suona talmente forte, talmente tanto, che l'uomo assume quello che Simmel [8a] definisce un "atteggiamento blasé", per il quale si potrebbe dire che l'uomo abbassa il volume di ciò che sente e la luminosità di ciò che vede in risposta al turbinio di stimoli che caratterizza la realtà urbana.
Jean-Paul Thibaud [10a] esplora un aspetto diverso del paesaggio sonoro urbano, concentrandosi sull'ascolto della musica attraverso le cuffiette mentre si cammina per le strade della città. Questo atteggiamento crea una connessione unica tra l'ascoltatore e l'ambiente circostante. Le cuffiette permettono alle persone di creare il proprio paesaggio sonoro, modificando la percezione dello spazio pubblico.
Ancora Bull, nel suo saggio sul paesaggio sonoro dell'automobile [9a], sottolinea come l'auto possa diventare uno spazio di performance e riflessione, spesso caratterizzato da una colonna sonora personalizzata. L'uso della musica all'interno dell'auto riflette l'umore e il momento del guidatore, contribuendo a definire la sua esperienza di guida e il rapporto con la città circostante.
In Brasile, nel quartiere di Petropolis, vengono evidenziati suoni che creano un'apparente sensazione di silenzio. Questi suoni, come gli uccellini o le voci dei bambini, sono sottolineati da Monica Fantini [11a], ma il silenzio è solo apparente, poiché i fili elettrificati emettono un costante ronzio che influisce sulla percezione dello spazio.
Infine, Xabier Erkizia [12a] esplora il paesaggio sonoro del treno, evidenziando come il suono del treno possa essere un elemento distintivo e identitario per le comunità. Il suono del treno può variare in base al tipo di treno e alla classe in cui ci si trova, influenzando l'esperienza dei passeggeri.
In generale, questi autori mettono in luce come i suoni urbani siano fondamentali per la comprensione e la definizione di un luogo. Il paesaggio sonoro urbano è in costante evoluzione e

rappresenta una parte essenziale della nostra vita quotidiana, influenzando il nostro rapporto con lo spazio e la comunità circostante.

2.1.4. Il rapporto col rumore

Nel contesto del rumore ambientale, si può distinguere tra paesaggi sonori "Hi-fi", caratterizzati da suoni chiaramente udibili e distinti, e paesaggi "Lo-fi", in cui i suoni sono sovrapposti e coperti da un rumore di fondo, comuni nelle società industrializzate. Il profilo acustico di questi suoni è fortemente influenzato dall'ambiente acustico circostante. L'inquinamento acustico è diventato un problema serio, soprattutto per i giovani, minando la loro salute e il processo di apprendimento.
Il rumore ha il potere di disturbare e omologare i suoni prodotti da apparecchi meccanici, contribuendo a rendere le città omologate e perdendo la loro unicità. Tuttavia, i suoni che causano l'inquinamento acustico sono parte integrante del paesaggio in cui viviamo. Una soluzione potrebbe essere riconoscere al rumore la stessa dignità estetica che attribuiamo al suono. Infatti, alcuni sostengono che il rumore stesso potrebbe diventare parte di un design del paesaggio sonoro, se affrontato con sensibilità e attenzione. Questo aspetto può essere visto da due prospettive: una scientifica, che mira a limitare il rumore ambientale, e una umanistica, che cerca di comprenderlo meglio [13a]. 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. Schäfer ha stimato che la battaglia tra l'espressione sonora e il controllo contribuisce ad aumentare i livelli sonori ambientali da 0.5 a 1dB l'anno, diventando quindi un vero e proprio generatore di rumore [26a] [Figura n.2.1].



[Figura n.2.1]

2.1.5. Come ‘catturare’ il paesaggio sonoro?

Nel discutere la differenza tra la vista e l'udito, molti autori sottolineano che il primo ha sviluppato mezzi per registrare e rappresentare ciò che percepiamo nel tempo, mentre il secondo non ha goduto di sviluppi simili. La registrazione dei suoni è diventata essenziale per catturare il paesaggio sonoro, consentendo l'analisi e la conservazione degli eventi sonori, simile a come le immagini vengono preservate nello spazio visivo. Negli ultimi dieci anni, si è assistito a un aumento significativo dei progetti di mappatura del paesaggio sonoro, utilizzati per comprendere i componenti sonori, i loro significati, e per controllare la struttura sonora. Un esempio di questa mappatura è la cartografia a isofone, con linee che racchiudono territori con lo stesso livello di decibel. Progetti come il "Ticino Soundmap" [1b] hanno esplorato il Canton Ticino, mappando i suoni distintivi della regione. Durante un corso di Didattica della Geografia, agli studenti è stato chiesto di esplorare i suoni geolocalizzati sulla mappa e utilizzare questi dati per creare nuove narrazioni, includendo elementi storici, culturali, e valori etici del territorio. Per una spiegazione più tecnica sulla cattura sonora si lascia al lettore la possibilità di approfondire l’argomento sulle linee guida rilasciate dal WWF [17b]. L'articolo fornisce una guida completa per il monitoraggio acustico della fauna selvatica, offrendo una panoramica dettagliata su come funzionano i sensori acustici, l'elaborazione dei segnali e l'analisi delle frequenze. Esamina anche le tendenze attuali e le limitazioni del monitoraggio acustico e fornisce consigli pratici su come scegliere i sensori acustici, pianificare ricerche, testare attrezzature e gestire i dati. L'articolo si rivolge agli ecologisti interessati al

monitoraggio del paesaggio sonoro naturale e fornisce una preziosa guida pratica. Non è l'obiettivo di questa tesi approfondire ulteriormente l'argomento.

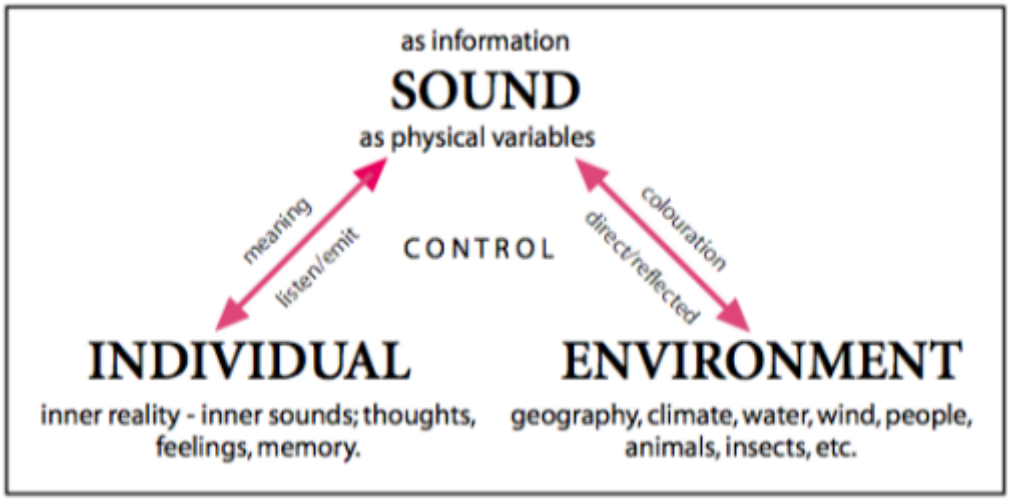
2.1.6. Esplorare il mondo attraverso i suoni

Cosa spinge a concentrarsi sul paesaggio sonoro e sull'attaccamento al luogo nella scuola? La risposta è più profonda di quanto si possa immaginare. Si cominci col dire che gli esseri umani non percepiscono il mondo solo attraverso la vista, ma attraverso tutti i sensi. Tuttavia, sembra che la vista abbia un'importanza sproporzionata rispetto agli altri sensi. Questo è un errore, e la scuola, sin dall'infanzia, dovrebbe insegnare ai bambini a dare importanza all'udito e a riscoprire la percezione multi-sensoriale del mondo. I suoni sono parte integrante della nostra cultura, ed è interessante notare che l'UNESCO li considera patrimonio immateriale. Questo patrimonio comprende sia gli elementi naturali, come fiumi e foreste, sia le strutture urbanistiche, come strade e stadi, e persino i suoni legati alla tradizione musicale e agli strumenti tipici di una regione. Inoltre, i suoni possono raccontarci molto di un luogo e delle attività che si svolgono in esso. I suoni possono aiutarci a conoscere meglio i luoghi e a entrare nel tessuto quotidiano della vita in quei posti. L'udito permette di percepire il mondo in modi che la vista da sola non può. Ad esempio, alcune persone ipovedenti si orientano utilizzando l'ecolocalizzazione, ascoltando il suono riflessi da superfici circostanti. Questo è un esempio di come l'udito possa aiutarci a navigare nello spazio in modo diverso dalla vista. Inoltre, i suoni possono rivelare la presenza di eventi o contesti prima che siano visibili. Ad esempio, i suoni di una festa all'aperto possono dirci che sta accadendo qualcosa, anche prima di vederla. Lavorare con i paesaggi sonori a scuola è stimolante e coinvolge gli studenti in attività che sviluppano il pensiero critico, la creatività e le strategie di apprendimento [20b]. Inoltre, aiuta gli studenti a sviluppare una maggiore consapevolezza ambientale. Lavorare sul concetto di attaccamento al luogo è altrettanto importante. Gli esseri umani hanno bisogno di sviluppare un legame con i luoghi per sentirsi al sicuro e per costruire relazioni significative. Inoltre, l'attaccamento al luogo porta ad un coinvolgimento attivo nella comunità e al desiderio di prendersi cura dell'ambiente circostante. Affrontare entrambe queste tematiche in classe può aiutare gli studenti a vivere realmente gli spazi che li circondano, sviluppando un forte legame con essi. Inoltre, promuove obiettivi di sviluppo sostenibile come la salute, l'istruzione di qualità, la sostenibilità e la partecipazione attiva nella comunità.
In sintesi, lavorare sul paesaggio sonoro e sull'attaccamento al luogo a scuola non solo rende l'apprendimento più coinvolgente, ma contribuisce anche a creare cittadini consapevoli, responsabili e attenti all'ambiente e alla comunità in cui vivono [17a].

2.2. RUOLO DEI SUONI AMBIENTALI

I suoni ambientali svolgono un ruolo significativo nell'ambiente naturale e nella vita umana. Essi costituiscono una forma di comunicazione naturale tra gli animali, aiutandoli a segnalare il territorio, attirare un partner o avvisare di pericoli imminenti. Inoltre, i suoni ambientali sono utilizzati come punti di riferimento dagli animali per orientarsi e individuare risorse vitali, come nel caso degli uccelli migratori che utilizzano il suono del mare per guidare il loro volo. Per gli esseri umani, i suoni ambientali hanno un impatto positivo sul benessere, riducendo lo stress, migliorando la concentrazione e promuovendo un senso di calma [22b]. Questi suoni possono anche contribuire all'identità e alla cultura di una regione o di una comunità, ispirando la musica tradizionale e altre forme d'arte. Inoltre, possono fornire informazioni sulla salute di un ecosistema: le variazioni nei modelli di suoni ambientali possono indicare cambiamenti nell'ambiente, come la deforestazione o l'inquinamento acustico. Inoltre, i suoni ambientali possono essere fonte di ispirazione per artisti, musicisti e creativi, che li utilizzano come base per creare opere d'arte, composizioni musicali e altre forme di espressione. Infine, i suoni ambientali sono importanti per la ricerca scientifica, poiché vengono utilizzati per studiare la biodiversità, la dinamica degli ecosistemi e gli effetti dei cambiamenti climatici.

In sintesi, i suoni ambientali sono una parte integrante dell'ecosistema e della vita umana, contribuendo in vari modi al benessere, alla comunicazione, all'identità culturale e all'ispirazione creativa [Figura n.2.2]. Pertanto, è cruciale preservarli e rispettarli per sostenere l'equilibrio ecologico e il benessere umano [16a].



[Figura n.2.2]

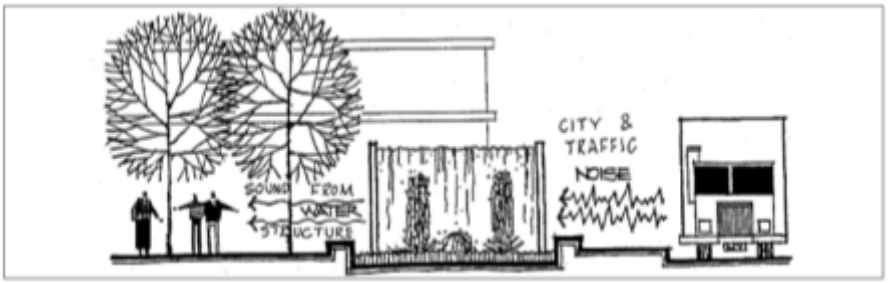
2.3. MINACCE ALLA CONSERVAZIONE DEI SUONI AMBIENTALI

I suoni ambientali, essenziali nell'ambiente naturale, affrontano diverse sfide e minacce alla loro preservazione [21b]. Uno dei principali pericoli è l'inquinamento acustico, causato da suoni indesiderati e spesso dannosi generati dalle attività umane, come il traffico stradale, ferroviario e aereo, l'industria, la costruzione e l'intrattenimento ad alto volume. L'urbanizzazione, come anche l'uso intensivo delle risorse naturali in alcune aree, inoltre, può alterare o distruggere habitat naturali caratteristici di determinati suoni ambientali, portando alla perdita di biodiversità e alla scomparsa di specie animali e vegetali [20a][19b]. Cambiamenti climatici e l'inquinamento delle acque danneggiano per esempio gli ecosistemi acquatici influenzando i suoni che provengono da fiumi, laghi e oceani [19b].
La conservazione dei suoni ambientali è importante non solo per il benessere degli ecosistemi, ma anche per la qualità della vita umana. Gli sforzi per mitigare queste minacce includono la regolamentazione del rumore, la conservazione degli habitat naturali, la promozione di pratiche sostenibili e la sensibilizzazione sul valore dei suoni ambientali e sulla necessità di proteggerli. La protezione di questi suoni è essenziale per preservare l'equilibrio tra la natura e la vita umana [3a] [4a].

2.4. METODI DI CONSERVAZIONE

La conservazione e la preservazione dei suoni ambientali richiede l'implementazione di diverse strategie e metodi. Le leggi e i regolamenti sull'inquinamento acustico limitano le emissioni di rumore e stabiliscono limiti di rumore accettabili in diverse aree, contribuendo a ridurre l'impatto dell'inquinamento acustico. Un altro aspetto importante è la conservazione degli habitat naturali. La creazione di parchi nazionali, riserve naturali e zone protette è fondamentale per mantenere gli ambienti naturali intatti. Inoltre, una pianificazione urbanistica o territoriale oculata può aiutare a mitigare l'inquinamento acustico, prevedendo la creazione di zone verdi e limitando l'espansione delle aree industriali o residenziali rumorose. L'educazione e la sensibilizzazione sono altrettanto cruciali. Campagne informative possono promuovere la consapevolezza sull'importanza dei suoni ambientali e incoraggiare il pubblico a ridurre il rumore e a rispettare gli ambienti naturali. Le tecnologie di riduzione del rumore vengono sviluppate per veicoli, macchinari industriali e infrastrutture di trasporto, contribuendo a ridurre l'impatto del rumore. Il monitoraggio costante dei suoni ambientali e la ricerca scientifica forniscono dati importanti per comprendere l'evoluzione degli ambienti sonori e sviluppare strategie di conservazione. Promuovere pratiche sostenibili in agricoltura, pesca e industria può contribuire a ridurre l'impatto dei suoni ambientali sugli ecosistemi. La pianificazione degli eventi e dell'intrattenimento pubblico è importante per ridurre il

rumore e limitare le distrazioni acustiche nelle aree sensibili. La creazione di zone acustiche [Figura n.2.3] può aiutare a regolare l'uso del suolo in modo da minimizzare l'interferenza tra attività rumorose e aree sensibili ai suoni ambientali. Infine, la collaborazione internazionale è spesso necessaria per affrontare il problema del rumore oltre le frontiere nazionali. In definitiva, l'obiettivo principale è trovare un equilibrio tra le attività umane e la protezione dei suoni naturali e dell'ambiente sonoro [2a] [4a] [18a].



[Figura n.2.3]

2.5. STUDI DI CASI

Lo studio e la conservazione dei paesaggi sonori ha molteplici implicazioni, sia per la conservazione dell'ambiente naturale e delle sue funzioni, sia per offrire al visitatore attento e consapevole il ristoro dato dal silenzio e da un ambiente ricco di stimoli con le continue variazioni dei suoni naturali. Attraverso le loro manifestazioni sonore il visitatore percepisce la vita degli animali che lo circondano, anche quando non visibili, e la ricchezza e biodiversità dell'ambiente naturale. I sentieri naturalistici possono anche essere percorsi di ascolto [19a] e contribuire alla formazione di una più ampia consapevolezza della ricchezza, diversità, e complessità degli ambienti naturali, ma anche della loro vulnerabilità di fronte al rumore prodotto dall'uomo.
Di seguito verranno presentati alcuni esempi che dimostrano l'importanza della conservazione e della preservazione dei suoni ambientali:
-Parco Nazionale di Yellowstone, USA [2b]: questo parco è un esempio di successo nella conservazione dei suoni ambientali in un ambiente naturale. Il parco ha implementato restrizioni sul rumore, limitando l'accesso alle aree più sensibili e promuovendo l'educazione sul silenzio. Queste misure hanno contribuito a preservare i suoni naturali del parco, inclusi il bramito degli alci e il canto degli uccelli.
-Santuario delle balene di Silver Bank, Repubblica Dominicana [3b]: in questo santuario marino, le barche sono regolamentate per evitare il disturbo delle balene durante il loro periodo di riproduzione. Queste misure hanno contribuito a preservare il suono delle canzoni delle balene megattere, che svolgono un ruolo essenziale nella comunicazione tra questi maestosi mammiferi marini.

-Parco Nazionale di Banff, Canada [4b]: questo parco ha affrontato la sfida dell'inquinamento acustico causato dal traffico stradale. È stato implementato un sistema di monitoraggio acustico per misurare i livelli di rumore e sono state apportate modifiche alla pianificazione dei percorsi per ridurre l'impatto sonoro sull'ambiente naturale.
-Riserva naturale del Deserto di Atacama, Cile [5b]: questa zona è famosa per essere una delle più silenziose al mondo, grazie alla sua posizione remota e all'altitudine. La conservazione del silenzio naturale è fondamentale per gli studi astronomici condotti nella regione, e sono state adottate misure per limitare l'inquinamento acustico.
-Progetto di reintroduzione dei licaoni, Sudafrica [6b]: i licaoni, o cani selvatici africani, sono noti per le loro vocalizzazioni complesse. In un progetto di conservazione, è stata prestata attenzione ai suoni emessi da questi cani per monitorare la loro presenza e il successo del programma di reintroduzione in un ambiente protetto.
-Fragment of Extinction [7b]: nato nel 2001 il progetto “Fragments of Extinction” ha l’obiettivo di andare ad indagare la complessità sonora del patrimonio acustico degli ecosistemi equatoriali nelle ultime aree incontaminate del pianeta, creando al contempo consapevolezza pubblica sul tema di quella che è stata definita come la sesta estinzione di massa.

Ma anche nel territorio nazionale si possono apprezzare ben 18 parchi nazionali e numerose riserve regionali.
Inoltre tra i progetti internazionali e non con lo scopo di preservare i suoni ambientali del territorio cito i seguenti:
-Quiet Parks International [8b]: questa organizzazione internazionale si impegna nella protezione di parchi e aree naturali con valori acustici eccezionali. Promuovono l'importanza del silenzio naturale e lavorano per garantire che le aree protette rimangano luoghi in cui si può sperimentare la quiete.
-National Park Service, USA [9b]: il National Park Service degli Stati Uniti è coinvolto in ricerche e monitoraggio dei suoni ambientali nei parchi nazionali. Questi studi contribuiscono alla gestione delle aree protette e all'attuazione di politiche di conservazione.
-European Soundscape Award [10b]: questo premio europeo è dedicato alla promozione e al riconoscimento delle iniziative di conservazione dei suoni ambientali in Europa. Il premio riconosce le migliori pratiche per la protezione del paesaggio sonoro.
-Progetto di mappatura acustica del suolo dell'Unione Europea [11b]: l'Unione Europea ha avviato un progetto di mappatura acustica per monitorare il rumore ambientale in tutto il continente. Questo progetto contribuisce all'identificazione delle aree che richiedono azioni di conservazione dei suoni ambientali.

-Progetto SABIOD (Scaled Acoustic Biodiversity) [12b] [13b]: delle Università di Pavia e di Tolone (Francia) per lo studio del paesaggio sonoro in ambienti naturali a diverso grado di integrità e di tutela, dalle riserve naturali integrali alle aree più soggette ad attività antropiche. Il progetto è anche collegato al tema delle "quiet areas" [15b] e all'importanza del silenzio, inteso come assenza del rumore delle attività umane, soprattutto dovute ai trasporti (traffico stradale, ferroviario e aereo), che ha un impatto negativo sulla salute umana [16b], sugli animali e sugli ecosistemi.
-Associazioni e ONG: numerose associazioni e organizzazioni non governative in tutto il mondo si concentrano sulla conservazione dei suoni ambientali e sulla sensibilizzazione del pubblico sull'importanza di questi suoni. Ad esempio, l'Archivio Italiano Paesaggi Sonori (AIPS) [14b] promuove la ricerca e la conservazione dei paesaggi sonori in Italia.

Questi studi di casi dimostrano come la conservazione dei suoni ambientali sia essenziale per la tutela degli ecosistemi, la ricerca scientifica e l'esperienza umana negli ambienti naturali. Proteggere i suoni ambientali contribuisce a preservare la biodiversità, promuovere la sostenibilità e migliorare la qualità della vita.

3. IL PROGETTO DEL MUSEO DI GALZIGNANO

3.1. OBIETTIVI

Realizzare una web app per fornire al visitatore un'esperienza digitale e interattiva nella “sala degli uccelli" del museo di Galzignano T. (PD). La web app gira su un totem con un pannello touch con cui il visitatore può interagire per selezionare l’habitat e l’uccello di cui vuole sentire il suono. La web app si collega ad un server a cui spetta l’interpretazione dei messaggi delle azioni utente e la riproduzione delle risorse audiovisive tramite un collegamento all’impianto di produzione della sala. Quindi il server oltre a collegarsi a degli speaker posti agli angoli della sala per riprodurre il suono, si collega ad un proiettore che proietta l’immagine dell’habitat selezionato su un plastico 3D. L'obiettivo è consentire all’utente di poter ascoltare più suoni di uccelli all'interno di uno stesso suono ambientale in modo da poter ricreare la sensazione di immersione acustica originale e fedele a quella che potrebbe assistere nell'habitat reale. La web app deve offrire all'utente la possibilità di:

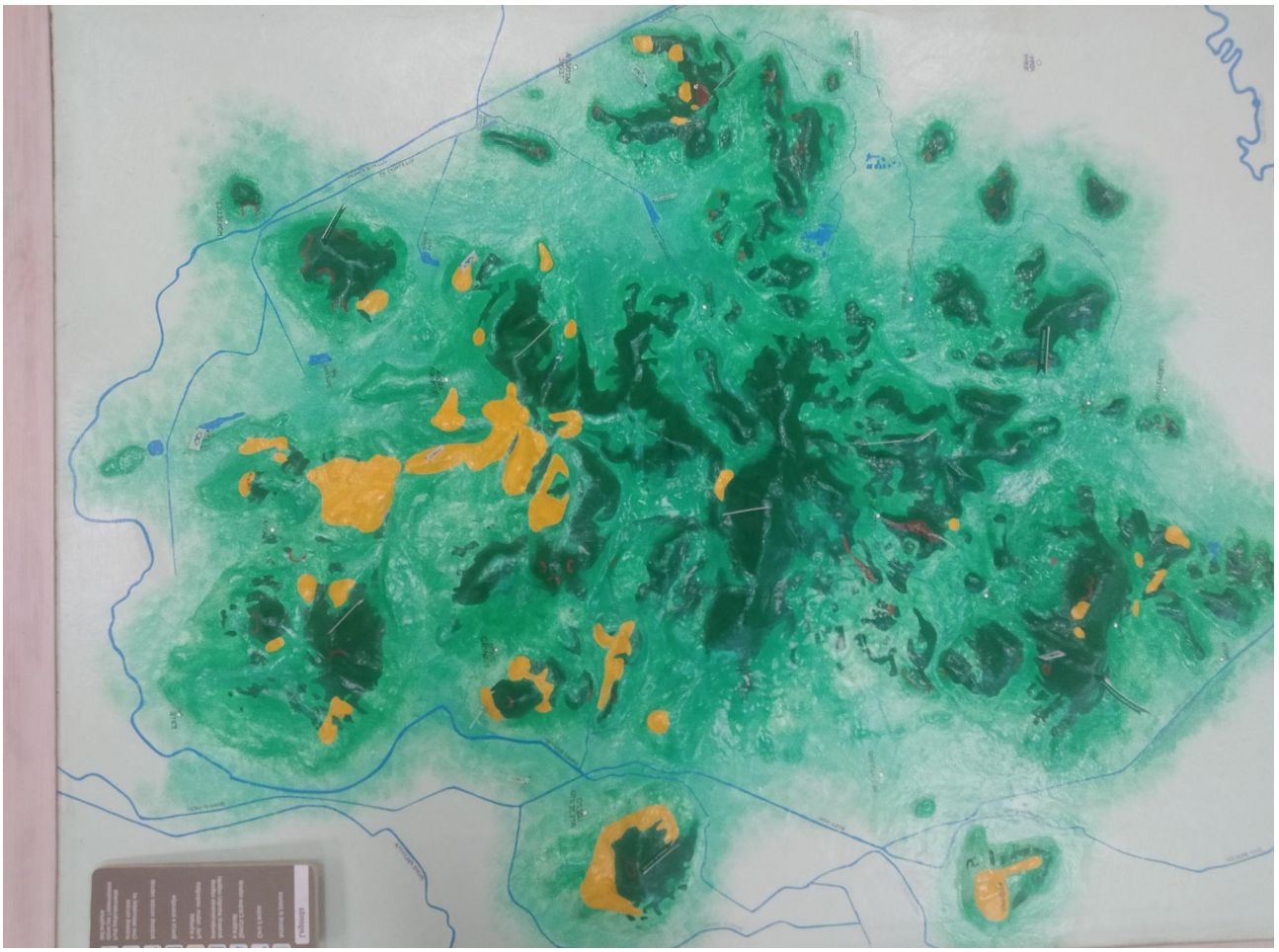
1) poter selezionare uno dei sei habitat alla volta
2) una volta scelto l’habitat deve iniziare sia la riproduzione del suono ambientale di sottofondo sia la proiezione della maschera territoriale sul plastico 3D
3) poter selezionare qualsiasi uccello presente in quell’habitat e sentirne la riproduzione sovrapposta dei canti
4) tornare indietro e fermare la riproduzione in qualsiasi momento.

La web app offre l’interfaccia grafica per l’interazione utente e si comporta da client, inviando messaggi in base alle selezioni dell’utente. Il server è un'applicazione a parte che elabora i messaggi e manda l’output corrispondente alle casse o al proiettore della sala.
Si ricorda che l'obiettivo di questa tesi riguarda esclusivamente la realizzazione del server, non della web app con cui il server deve interagire. Il web server in questo progetto non è alla versione finale poiché non si interfaccia ancora con la web app per l’invio dei messaggi di stato, utili alla web app per fornire un feedback visivo all’utente.

3.2. REALIZZAZIONE E COMPONENTI

La web app è stata realizzata in Java e ha principalmente il compito di fornire all’utente un’interfaccia grafica con cui poter interagire per selezionare l’oggetto di proprio interesse.
Il componente che si occupa dell’elaborazione delle azioni dell’utente sulla web app, della riproduzione audio del suono e dell'immagine dell’habitat è invece il server web.

Il server web è stato realizzato con Processing (basato su Java) ed è strutturato per ricevere messaggi da un client (web app), connesso alla stessa porta e IP, per la connessione si usa il protocollo WebSocket. Ad ogni messaggio del client corrisponde un file audio per il canto degli uccelli, e, nel caso di un habitat, un'immagine da proiettare sul plastico 3D della sala [Figura n.3.1]. L'immagine è in risoluzione FHD (1920x1080), formato .png e in bianco e nero, mentre il file audio è in formato .mav.



[Figura n.3.1]

Ogni file audio di uccello ha la durata di massimo 10 secondi mentre il file audio ambientale dura dai 2 ai 4 minuti. È possibile riprodurre più file audio del canto degli uccelli, anche sovrapposti, assieme all'audio ambientale dell'habitat selezionato, all'interno della stessa regione abitativa; quando tuttavia si cambia habitat, l'audio, dell'ambiente e di eventuali uccelli in riproduzione, come la proiezione della maschera dell'habitat precedente, se presente, vengono cancellate per evitare noiose sovrapposizioni. Il server implementa anche la gestione del messaggio "stop". Questo messaggio una volta ricevuto cancella qualsiasi riproduzione sonora e visiva in corso. È utile nel

caso l'utente volesse interrompere prima della fine l'ascolto del soundscape per passare ad un altro ambiente o ricrearne uno da capo.
Le tre funzioni principali del server sono:

- `void setup() {...}` : dove vengono inizializzate le variabili globali, invocato il costruttore `WebsocketServer` per inizializzare la variabile `ws`, e assegnati gli elementi chiave/valore della mappa `suoniAmbiente`, per la riproduzione del suono ambientale in background.
- `void draw() {...}` : dove indicato il colore di background dell'immagine proiettata, in questo caso settato su nero in quanto le immagini sono tutte .png senza sfondo in trasparenza, e gestita la proiezione.
- `void webSocketServerEvent(String msg) {...}` : dove vengono gestiti i messaggi del client a seconda che siano "stop", di ambienti o di uccelli.

3.3. PRODOTTO FINALE

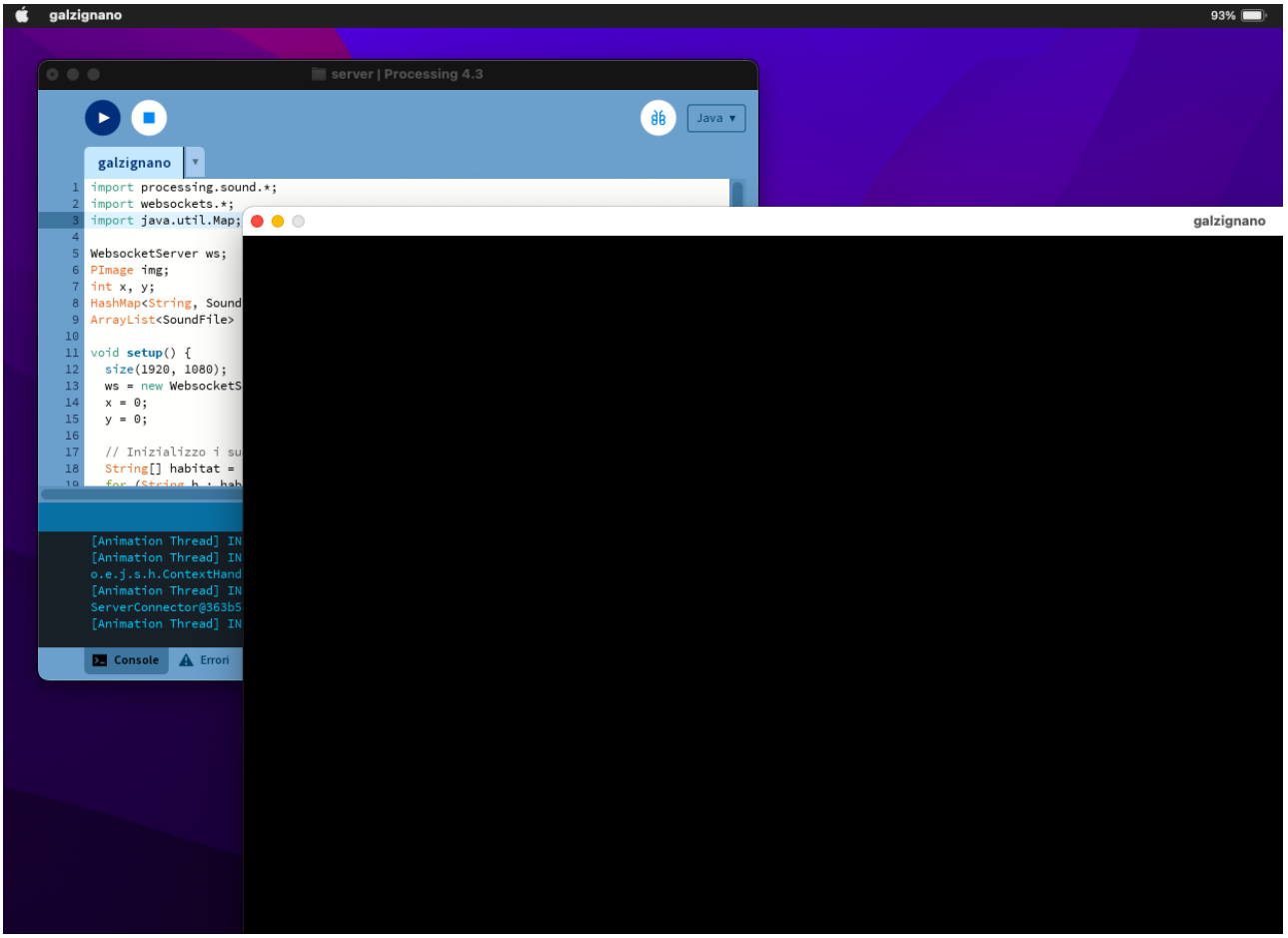
Il prodotto finale rispecchia tutte le caratteristiche fornite dal cliente. È quindi un server in Java che si connette ad un client di prova su una connessione WebSocket all'URI `ws://localhost:8025/john`, all'indirizzo IP `localhost` e alla porta `8025`, su un canale all'interno del server chiamato `john`. Il client invia messaggi di tipo `String` al server che li legge e in base a cosa corrispondono, "stop", nome di un uccello o nome di un habitat, riproduce l'audio corrispondente. La corrispondenza tra il messaggio e i file audio (.wav) e/o immagine (.png) è stata implementata adottando il seguente standard nomenclativo: i messaggi che possono essere inviati dal client sono uguali al nome dei file audio e dei file immagine che sono memorizzati nella stessa directory del server nella cartella "data" a cui viene aggiunto il suffisso riguardante il tipo (come detto prima .wav per gli audio e .png per le immagini). Facciamo un esempio: per l' habitat `"Antropici"` si ha il file audio `Antropici.wav` e il file immagine `Antropici.png` , il messaggio che viene inviato dal client al server quando l'utente seleziona l'habitat antropico è `"Antropici"`; nel caso degli uccelli quando si seleziona l' uccello `"Ghiandaia"` il messaggio inviato dal client al server è `"Ghiandaia"` a cui corrisponde solo il file audio `Ghiandaia.wav`.
Lo standard adottato per la nomenclatura dei nomi degli uccelli, adottato sia nei messaggi del client sia per i file nella cartella "data" è: iniziale maiuscola e il resto minuscolo, se il nome è composto le parti sono separate da un trattino senza spazi e tutto minuscolo e eventuali apostrofi o accenti non devono essere scritti (es. `Passera-d-italia.wav`). Nel caso degli habitat quindi oltre al soundscape corrispondente viene proiettata un'immagine sul plastico 3D nella sala del museo, si noti che a un habitat corrisponde un solo suono ed una sola mappa, per ogni uccello un solo suono

(non è importante la risoluzione dell'immagine dell'habitat in quanto questa viene riscalata sempre in 1920x1080 ovvero la risoluzione del proiettore con `img.resize(1920, 1080);` utile in ottica di scalabilità in quanto permette, in futuro, di aggiungere nuovi habitat con le relative immagini .png a qualsiasi risoluzione). In particolare si usa una `HashMap<String, SoundFile> suoniAmbiente = new HashMap<String, SoundFile>()` per mappare i suoni ambientali e le rispettive mappe da proiettare e una `ArrayList<SoundFile> suoniUccelliInRiproduzione = new ArrayList<SoundFile>()` per tenere traccia dei suoni degli uccelli in riproduzione, visto che si vuole rendere possibile la sovrapposizione dei canti degli uccelli ma non quella dei suoni ambientali. Per realizzare ciò inoltre quando il server riceve un messaggio corrispondente ad un suono ambientale ferma e cancella la riproduzione di tutti gli audio sia ambientali che del canto degli uccelli in corso per riprodurre solamente l'ultimo audio ambientale ricevuto. Come già detto è stato anche implementato un messaggio `"stop"` che ferma la riproduzione di tutti i suoni, ambientali e animali. Il messaggio potrebbe corrispondere al tasto "indietro" nella web app (non è stato ancora deciso definitivamente al momento della realizzazione del server anche se ne è stata suggerita l'implementazione), utile se l'utente sbaglia a selezionare un habitat o non vuole aspettare che finisca la riproduzione del suono ambientale per cambiare scenario. Si noti che la mappatura habitat-uccello non è 1-1, in altre parole un uccello può essere presente in più habitat. Tuttavia non è stato necessario gestire questi casi dal server in quanto sarà l'interfaccia utente della web app che mostrerà all'utente tutti e soli gli uccelli esistenti nell'ambiente selezionato dell'utente e, una volta selezionato la specie di cui si vuole sentire il suono, il client manderà sempre lo stesso messaggio al server che quindi, indipendentemente da in che habitat si trova l'uccello, riprodurrà il suo canto. Sarà però compito degli sviluppatori della web app mettere selezionabili gli uccelli giusti per ogni habitat. Si ricorda che il client è solo di prova quindi non è completo, nel senso che all'interno della funzione `String getCustomMessage(String input)` in `switch(input)` non sono presenti tutti i nomi del centinaio di uccelli della sala ma solo alcuni nomi, assieme a quegli degli habitat, necessari per testare il corretto funzionamento del server. Per client di prova si vuole dire che il client web, di cui nei prossimi paragrafi verrà illustrata l'implementazione, non sarà necessariamente il client finale con cui interagirà il server nella web app. Anche se si fa notare che il client finale dovrà comportarsi come quest'ultimo per poter interagire al meglio col server creato.

Si veda [5. APPENDICE] per il codice del server e del client completo.

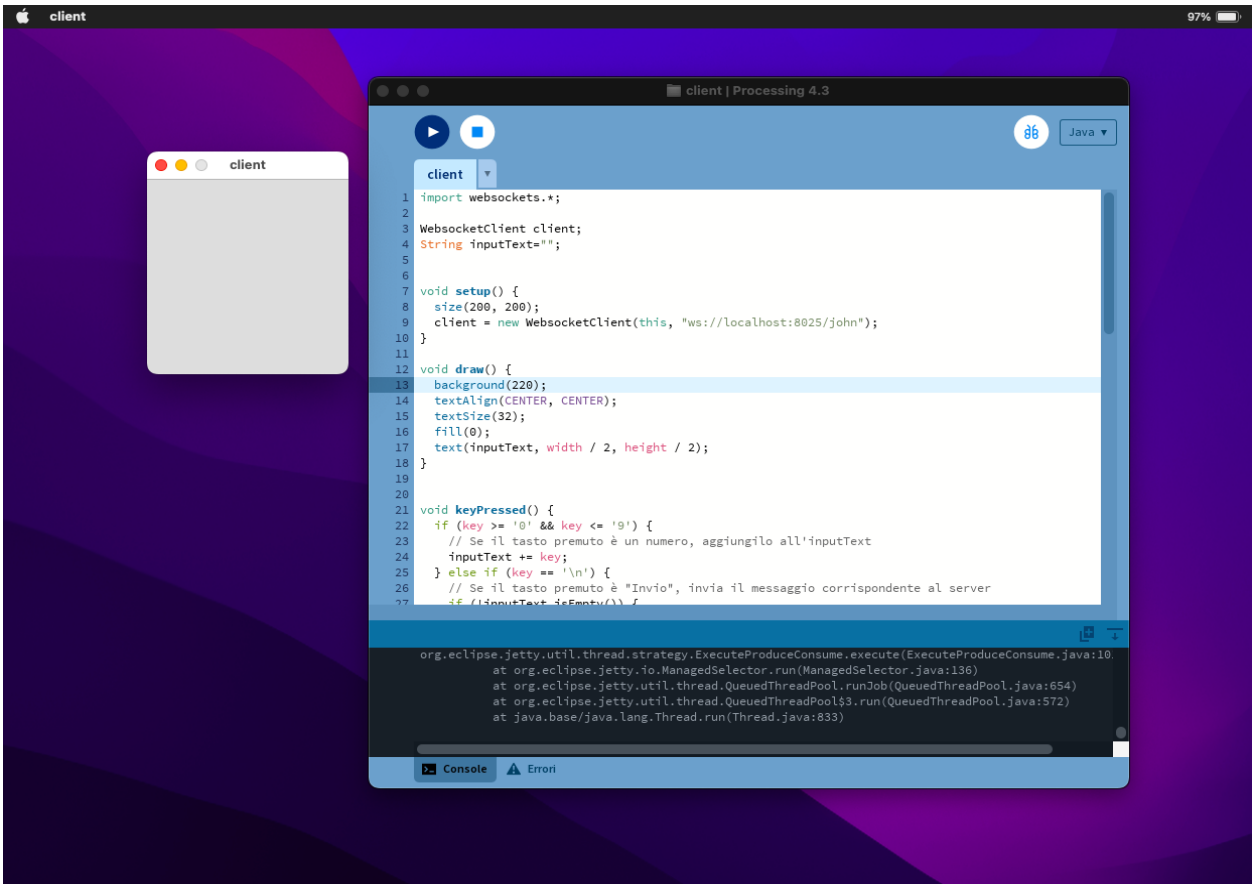
Seguono degli screen esemplificativi per mostrare il client e il server in funzione:

Avvio del server



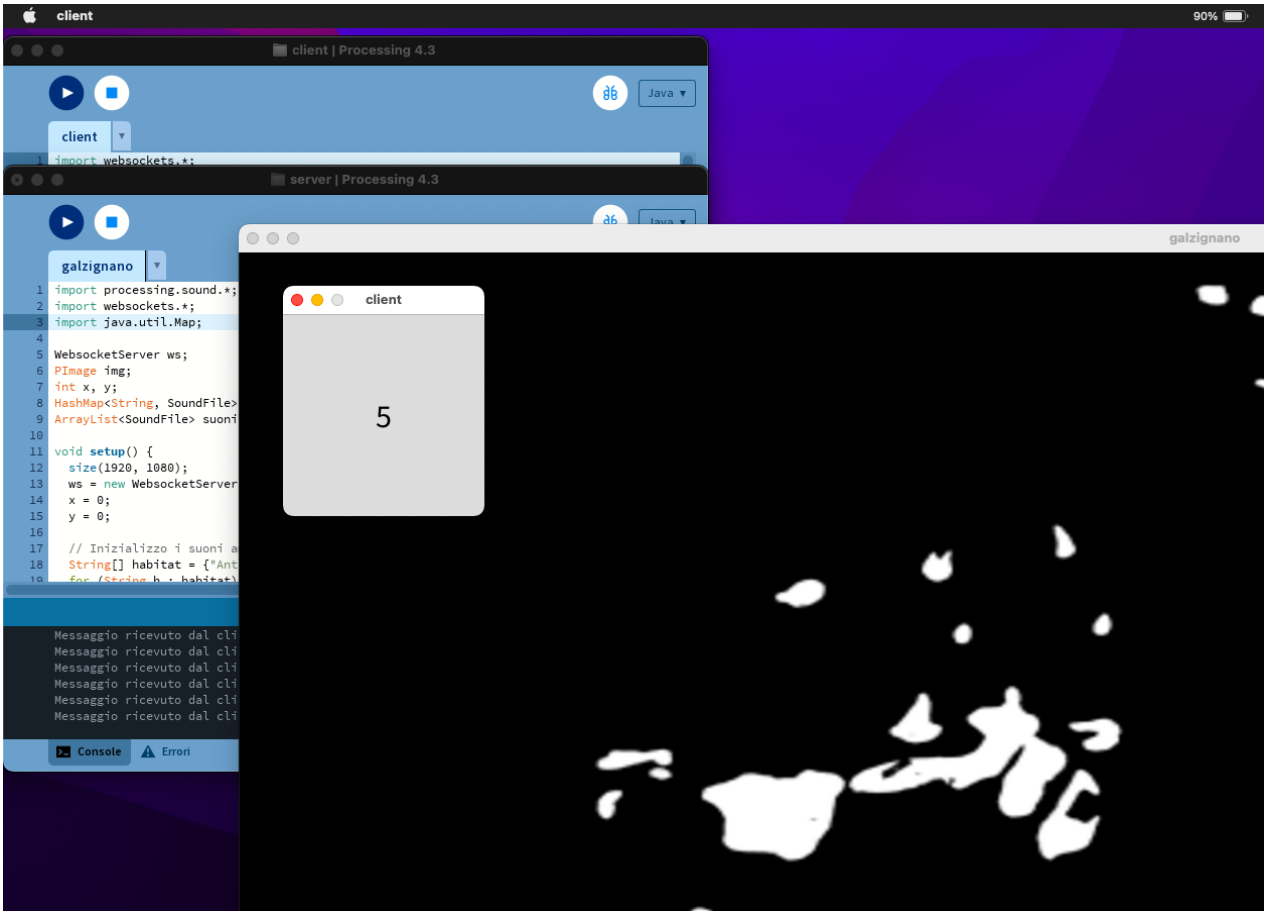
[Figura n.3.2]

Avvio del client



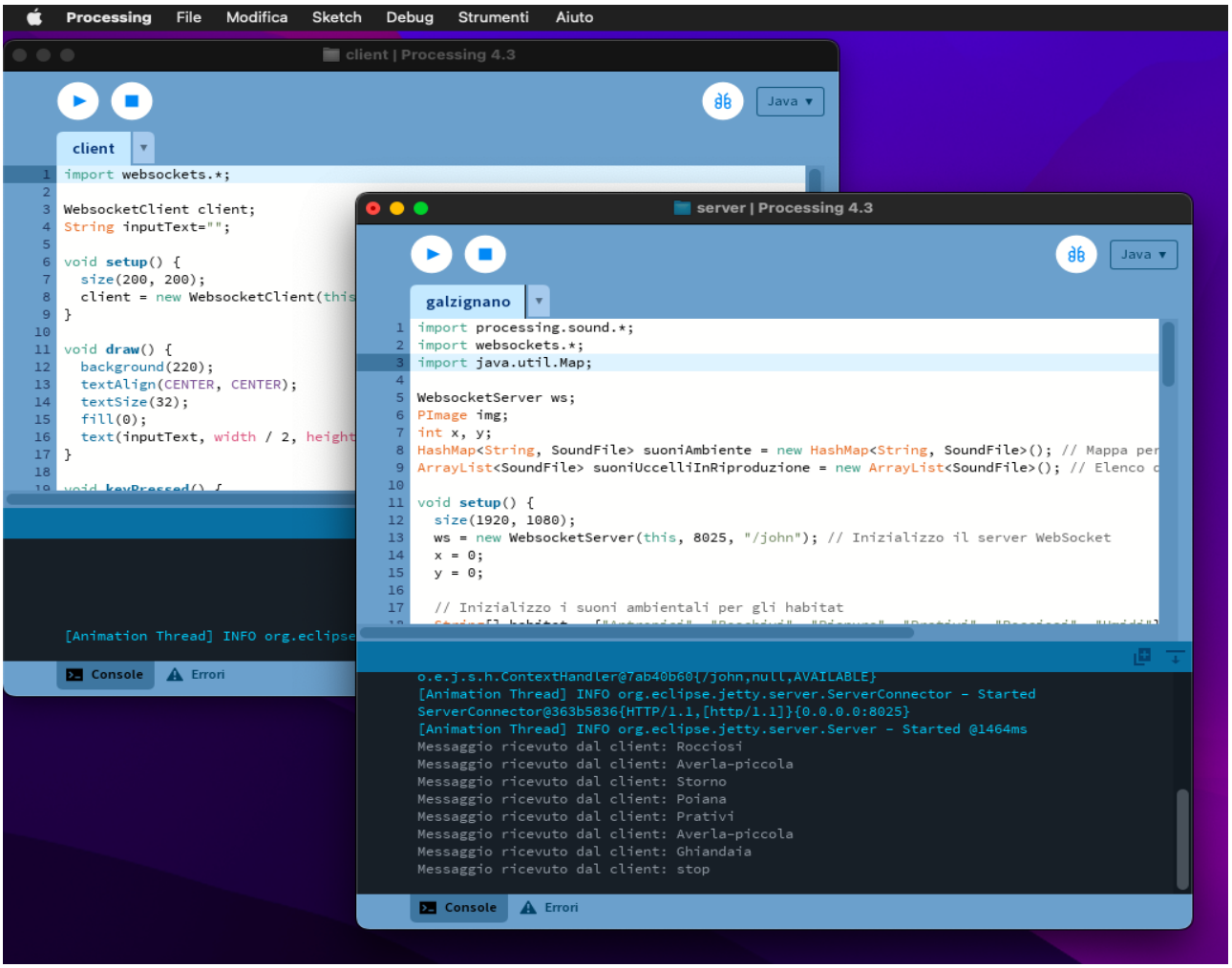
[Figura n.3.3]

Invio dei messaggi



[Figura n.3.4]

Chiusura



[Figura n.3.5]

4. PROGETTO E SVILUPPO DEL WEB SERVER

4.2. FONDAMENTI DI WEB SERVER E WEB APPLICATION

4.2.1. Concetti di base sui web server

I web server svolgono un ruolo fondamentale nell'ecosistema di Internet, consentendo la distribuzione di contenuti web su richiesta degli utenti. Questi server sono il cuore pulsante di siti web, applicazioni online e servizi che utilizziamo quotidianamente. In questo testo, esploreremo il funzionamento, le caratteristiche e l'uso dei web server.
Un web server è un software o un'infrastruttura hardware responsabile di ricevere, elaborare e rispondere alle richieste HTTP (Hypertext Transfer Protocol), o altri protocolli, provenienti da client, solitamente browser web. Il funzionamento generico di un web server si compone di:

1) Ricezione delle richieste: il web server ascolta costantemente le richieste in arrivo sulla porta specifica (di solito la porta 80 per HTTP e la porta 443 per HTTPS) del protocollo TCP/IP. Queste richieste possono provenire da browser, applicazioni o altri server.

2) Elaborazione delle richieste: una volta ricevuta una richiesta, il web server la decodifica e determina quale risorsa o servizio richiesto deve essere restituito al client. Questo può includere la ricerca di file, l'esecuzione di script o il reindirizzamento a un'altra URL.

3) Recupero dei dati: il web server recupera il contenuto richiesto dalla memoria o dal file system del server. Questo può essere un file HTML, una pagina dinamica generata da un'applicazione web o un file multimediale.

4) Generazione della risposta: il web server compone una risposta HTTP che include l'header di risposta (contenente informazioni sul server, sulla data e sul tipo di contenuto) e il corpo della risposta (il contenuto effettivo richiesto).

5) Invio della risposta al client: il web server invia la risposta al client tramite il protocollo HTTP. Il client, di solito un browser web, interpreta la risposta e visualizza il contenuto richiesto all'utente.

I web server possono variare notevolmente nelle loro caratteristiche e capacità, ma ci sono alcune funzionalità comuni che possiamo trovare in molti di essi:

1) Supporto per diversi protocolli: oltre a HTTP, i web server possono supportare protocolli come HTTPS, FTP, SPDY, WebSocket e altri per soddisfare una varietà di esigenze.

2) Gestione delle connessioni: i web server devono essere in grado di gestire molte connessioni simultanee in modo efficiente, garantendo tempi di risposta rapidi.

3) Sicurezza: i web server spesso offrono funzionalità di sicurezza, come la crittografia SSL/TLS, per proteggere le comunicazioni tra il client e il server.

4) Configurabilità: gli amministratori di sistema possono configurare il web server in base alle esigenze specifiche, regolando parametri come le autorizzazioni, le reindirizzazioni e le cache.

5) Log e monitoraggio: i web server registrano dettagli sulle richieste e le attività di accesso per scopi di monitoraggio, analisi e sicurezza.

6) Supporto per applicazioni dinamiche: alcuni web server supportano l'esecuzione di applicazioni web dinamiche, consentendo la generazione di contenuti in tempo reale attraverso script e framework come PHP, Ruby on Rails e Node.js.

I web server hanno molteplici utilizzi e applicazioni. I principali scenari in cui vengono utilizzati sono per esempio hosting di siti web, e-commerce o per applicazioni web.

Nello specifico nel progetto realizzato viene usato un protocollo WebSocket, un protocollo di comunicazione bidirezionale a bassa latenza che differisce significativamente da HTTP in quanto è progettato per supportare interazioni in tempo reale e connessioni persistenti. Ecco alcune delle principali differenze tra WebSocket e HTTP:

1) Comunicazione bidirezionale: mentre HTTP è un protocollo di comunicazione unidirezionale in cui il client invia una richiesta e il server risponde, WebSocket permette una comunicazione bidirezionale. Entrambi il client e il server possono inviare messaggi

l'uno all'altro in qualsiasi momento, senza dover attendere una richiesta specifica per mandare il messaggio.

2) Connessioni persistenti: in HTTP, ogni richiesta/risposta è una transazione indipendente, e dopo che il server ha risposto, la connessione viene chiusa. Con WebSocket, una volta stabilita la connessione iniziale, questa rimane aperta per l'intera durata della sessione, consentendo un flusso continuo di dati in entrambe le direzioni.

3) Basso overhead: WebSocket riduce l'overhead rispetto a HTTP. Le richieste HTTP includono spesso intestazioni (headers) aggiuntive e richiedono l'apertura e la chiusura di connessioni multiple. WebSocket utilizza un frame più leggero, il che lo rende più efficiente per la comunicazione in tempo reale.

4) Latenza Ridotta: a causa della sua connessione persistente e del minore overhead, WebSocket è noto per ridurre la latenza rispetto a HTTP, rendendolo ideale per applicazioni in tempo reale come chat, giochi online e strumenti di collaborazione.

5) Scalabilità: WebSocket è progettato per essere altamente scalabile, consentendo a un grande numero di client di connettersi a un server WebSocket senza un eccessivo aumento del carico di sistema rispetto a HTTP.

6) Protocollo di handshake: WebSocket richiede un handshake iniziale per stabilire la connessione, ma una volta stabilita, le successive comunicazioni sono leggere e veloci. Questo è diverso da HTTP, che richiede un handshake completo per ciascuna richiesta/risposta.

In termini di sicurezza i due protocolli sono simili: WebSocket può essere utilizzato sia su connessioni non sicure (ws://) che su connessioni sicure (wss://), come HTTP (http:// e https://).
In sintesi, WebSocket è una scelta ideale per applicazioni in tempo reale che richiedono comunicazioni bidirezionali a bassa latenza, come chat, giochi online, aggiornamenti in tempo reale e strumenti di collaborazione. HTTP, d'altra parte, è ottimo per richieste e risposte in un modello di comunicazione unidirezionale, come il recupero di pagine web statiche o dati da un server.
WebSocket è supportato dalla maggior parte dei browser moderni e da molte librerie di sviluppo, il che lo rende facilmente accessibile per sviluppatori di applicazioni web e mobile.

In conclusione, i web server sono fondamentali per il funzionamento di Internet e la distribuzione di contenuti web. Sono strumenti altamente configurabili e versatili che soddisfano una vasta gamma di esigenze, dal semplice hosting di siti web alla gestione di applicazioni web complesse e di servizi di rete. La loro continua evoluzione e la crescente attenzione alla sicurezza li rendono uno dei pilastri dell'era digitale [25a].

4.2.2. Tipi di web server

I web server possono variare in base al software utilizzato e alle loro funzionalità specifiche. I tipi di web server possono includere server web open source come Apache e Nginx, server web proprietari come Microsoft Internet Information Services (IIS) e server web specializzati progettati per scopi specifici. Questi server possono differire nella loro capacità di gestire il traffico, nella sicurezza, nell'estendibilità e in altre caratteristiche [25a].

4.2.3. Caratteristiche delle web application

Le web application sono applicazioni software distribuite tramite un web server e accessibili tramite un browser web o un'applicazione client. Le loro caratteristiche comprendono interattività, dinamicità e accesso ai dati. Inoltre, le web application spesso incorporano logica lato server per gestire l'elaborazione dei dati e la comunicazione con server di database o servizi web esterni. Le moderne web application sono progettate con l'obiettivo di fornire un'esperienza utente intuitiva, utilizzando tecnologie come JavaScript, HTML5 e CSS3 [25a].

4.2.4. Ruolo del web server nell'interfacciamento con la web app

Il web server svolge un ruolo cruciale nell'interfacciamento tra il client e la web application. Agisce come intermediario, accettando richieste HTTP dal client, instradandole alla web application corrispondente e restituendo le risposte al client. Nel processo, il server può gestire aspetti come autenticazione utente, gestione delle sessioni, caching e sicurezza. Inoltre, il web server è responsabile dell'analisi delle richieste, della comunicazione con il back-end dell'applicazione e della consegna di risposte ottimizzate per il client, spesso in termini di contenuto e formato.

Questo capitolo rappresenta una base tecnica essenziale per la comprensione dei principi fondamentali dei web server e delle web application, preparando il terreno per l'approfondimento nelle successive sezioni della tesi, in cui si affronteranno aspetti più avanzati e specifici dell'architettura e dello sviluppo di sistemi web [25a].

4.3. PROGETTAZIONE WEB SERVER

4.3.1. Requisiti di progettazione

Le funzionalità richieste in fase di sviluppo del server sono state discusse in [3.1 OBIETTIVI]. A queste si aggiungono:

- scalabilità e manutenibilità dell'applicazione (si veda capitolo [4.3.5.])
- affidabilità dell'applicazione (si veda capitolo [4.7.3.])
- assenza di un'interfaccia per l'interazione utente (il server deve svolgere autonomamente la gestione dei messaggi ricevuti dal client)
- testing (si veda capitolo [4.7.])

Non è stato espresso nessun requisito riguardo requisiti minimi di sistema o meccanismi di autenticazione avanzati per la sicurezza del server; a riguardo di ciò il controllo degli accessi base, offerto dalla libreria websockets, è stato sufficiente (si veda capitolo [4.3.4.] per maggiori dettagli). Inoltre non sono stati dati vincoli prestazionali o tempi di risposta massimi da rispettare.

Per questo progetto è stato utilizzato l'ambiente di sviluppo Processing nella versione 4.3. Per eseguire il codice realizzato non occorre altro che importare le seguenti librerie:

```
import processing.sound.*;
import websockets.*;
```

Per eseguire questo codice su Processing bisogna:

1. Aprire l'ambiente di sviluppo di Processing.
2. Incollare il codice nell'editor di Processing.
3. Premere il pulsante "Run" per eseguire il tuo server.

Per eseguire il server Processing realizzato, si consiglia di rispettare i requisiti di sistema minimi definiti come segue:

1. Memoria RAM: si consiglia almeno 512 MB di RAM per l'esecuzione del server. Questo dovrebbe essere sufficiente per gestire le operazioni di base, inclusa la riproduzione dei suoni e la visualizzazione delle immagini. Tuttavia, se si prevede di aumentare il numero di connessioni simultanee o di operare con file multimediali di dimensioni considerevoli, è consigliabile avere almeno 1 GB di RAM.

2. Processore (CPU): un processore dual-core con una frequenza di clock di almeno 2.0 GHz dovrebbe essere sufficiente per le operazioni del server. Questa configurazione dovrebbe gestire la riproduzione dei suoni e la visualizzazione delle immagini in modo adeguato. Se il carico di lavoro aumenta, un processore più potente potrebbe essere preferibile.

3. Spazio su disco: almeno 200 MB di spazio libero su disco dovrebbero essere sufficienti per il codice del server, le immagini e i file audio. Tuttavia, la quantità di spazio necessaria aumenterà con la dimensione dei file multimediali che si intende gestire.

4. Connessione Internet: se il server sarà accessibile online, è necessaria una connessione Internet stabile. Inoltre, ci si assicuri di configurare correttamente le regole del firewall e la gestione delle porte, poiché il server utilizza il protocollo WebSocket sulla porta 8025.

Queste sono stime conservative basate sul codice realizzato, e le esigenze specifiche possono variare se si intende implementare le funzionalità del server. In tal caso è consigliabile monitorare le risorse del sistema durante l'esecuzione del server per valutare eventuali necessità di ottimizzazione o di risorse aggiuntive.

4.3.2. Scelte di architettura

L'architettura del server è basata su WebSocket, un protocollo di comunicazione bidirezionale che consente una connessione persistente tra il server e il client. Questa scelta è stata fatta considerando i requisiti di tempo reale dell'applicazione, poiché WebSocket offre una latenza molto ridotta rispetto ai protocolli di comunicazione HTTP tradizionali ed è quindi generalmente consigliabile in questo tipo di applicazioni.
Il server utilizza la libreria "Websockets" (`import websockets.*;`) di Processing per implementare il protocollo WebSocket. La libreria semplifica la gestione dei socket, consentendo al server di ascoltare su una porta specifica e di stabilire connessioni WebSocket con i client.

```
import websockets.*;
WebsocketServer ws;
...
void setup() {
...
ws = new WebsocketServer(this, 8025, "/john"); // Inizializzo il server WebSocket
...
```

```
}
```

Questo approccio garantisce una comunicazione efficiente e sincrona tra il server e il client.
In merito a ciò si parlerà del funzionamento interno del server e come gestisce le connessioni sia HTTP che WebSocket. Le scelte di architettura riguardano la progettazione e l'implementazione del sistema, quindi è un contesto appropriato per spiegare il server Jetty.
Le scritte in blu visualizzabili nella console del server quando si avvia il programma sono messaggi di log generati dal server Jetty, che è utilizzato da Processing per gestire le connessioni WebSocket tramite la libreria Websockets. Ecco una spiegazione di questi messaggi:

1) `[Animation Thread] INFO org.eclipse.jetty.util.log - Logging initialized @1749ms`: questo messaggio indica che il sistema di logging di Jetty è stato inizializzato. Il valore numerico (in questo caso, `@1749ms`) indica il tempo impiegato per l'inizializzazione.

2) `[Animation Thread] INFO org.eclipse.jetty.server.Server - jetty-9.3.6.v20151106`: questo messaggio fornisce informazioni sulla versione di Jetty che viene utilizzata.

3) `[Animation Thread] INFO org.eclipse.jetty.server.handler.ContextHandler - Started o.e.j.s.h.ContextHandler@56e2e30{/john,null,AVAILABLE}`: questo messaggio indica che un gestore di contesto (context handler) è stato avviato con il nome '`/john`'. I contesti sono utilizzati per associare i percorsi delle richieste ai gestori appropriati.

4) `[Animation Thread] INFO org.eclipse.jetty.server.ServerConnector - Started ServerConnector@3c34538c{HTTP/1.1,[http/1.1]}{0.0.0.0:8025}`: questo messaggio indica che il server ha avviato un connettore di tipo '`HTTP/1.1`' che ascolta su '`0.0.0.0`' alla porta '`8025`'.

5) `[Animation Thread] INFO org.eclipse.jetty.server.Server - Started @2176ms`: questo messaggio indica che il server Jetty è stato completamente avviato. Il valore numerico (in questo caso, '`@2176ms`') indica il tempo impiegato per avviare il server.

Questi messaggi di log sono utili per il monitoraggio e il debug del server Jetty e possono aiutare a identificare eventuali problemi o a tenere traccia delle prestazioni del server durante l'esecuzione del programma. Si ricorda che possono altresì essere ignorati o disattivati se non sono necessari.
I messaggi che mostrano l'avvio di un connettore HTTP/1.1 possono essere fuorvianti quando si utilizza il protocollo WebSocket. In realtà, quando si avvia un server WebSocket tramite Jetty, non si sta avviando un server HTTP tradizionale, ma un server che può gestire anche il protocollo WebSocket. Quindi il termine "HTTP/1.1" nei messaggi di log potrebbe essere fuorviante. Il server WebSocket si basa su HTTP per stabilire la connessione iniziale (handshake) e, una volta stabilita la connessione WebSocket, il protocollo HTTP viene messo da parte per comunicare tramite WebSocket. Quindi, il server WebSocket può essere utilizzato per gestire sia richieste HTTP che connessioni WebSocket. Per riassumere, se si sta utilizzando un protocollo WebSocket nel programma, i messaggi riguardanti l'avvio del connettore HTTP/1.1 sono inerenti al funzionamento interno del server WebSocket basato su Jetty, che può gestire sia HTTP che WebSocket. Nonostante questi messaggi di log facciano riferimento a HTTP/1.1, il server sarà in grado di gestire correttamente le connessioni WebSocket.

Inoltre si è scelto di standardizzare la tipologia di file multimediali, optando per l' estensione .wav per i file audio e .png per i file immagine. La scelta non è casuale, infatti:

- nel caso del .png (Portable Network Graphics) è un formato di immagine raster ampiamente utilizzato che si distingue da altri tipi di formati di immagine per il fatto che utilizza una tecnica di compressione senza perdita, il che significa che l'immagine compressa può essere ripristinata all'originale senza alcuna perdita di qualità. Inoltre supporta la trasparenza, consentendo la creazione di immagini con sfondi trasparenti, ideale per proiettare la maschera di una mappa su un grafico 3D come nel caso in questione. Il formato PNG è in grado di gestire immagini a 24-bit di profondità di colore (16,7 milioni di colori), il che lo rende adatto per la rappresentazione di immagini fotografiche ad alta qualità.
- nel caso del .wav si è preferito adottarlo a discapito di un formato più comune come l'MP3 per diversi motivi, il primo è che contengono audio non compresso e quindi è un formato audio qualitativamente più alto e ad alta fedeltà. I suoni sono riprodotti senza perdita di qualità e con una gamma dinamica completa. Visto che è un formato senza perdita conserva tutti i dati audio originali. Inoltre poiché i file WAV sono senza perdita e non richiedono decodifica durante la riproduzione, possono offrire una latenza inferiore rispetto ai file MP3 in applicazioni audio in tempo reale.

4.3.3. Linguaggi di programmazione e framework per lo sviluppo del web server

Il server è stato sviluppato utilizzando il linguaggio di programmazione Processing. Processing è una scelta adatta per lo sviluppo di applicazioni interattive e visive, in quanto semplifica la gestione di grafica e input utente. Inoltre, Processing supporta l'uso di librerie esterne, come la libreria "Websockets" che fornisce un'implementazione robusta del protocollo WebSocket.
La libreria "Websockets" offre una serie di funzionalità per la gestione delle connessioni WebSocket, inclusa la gestione dei messaggi in arrivo e in uscita, consentendo al server di comunicare con il client in modo efficiente seguendo le specifiche al seguente link GitHub (https://github.com/alexandrainst/processing_websockets).

4.3.4. Sicurezza e gestione degli accessi

La sicurezza è una priorità nella progettazione del server. Per garantire la sicurezza, il server accetta connessioni solo sulla porta `8025` e richiede una chiave utente specifica (`"/john"`) per stabilire la connessione. Questo approccio riduce il rischio di accesso non autorizzato e limita le connessioni solo a client autorizzati. Per ulteriori livelli di sicurezza, è possibile implementare meccanismi di autenticazione e autorizzazione più avanzati, come l'uso di token di accesso o certificati SSL, a seconda dei requisiti specifici del progetto; in questo caso non è stata richiesta l'implementazione di alcun livello di sicurezza aggiuntivo.

4.3.5. Scalabilità e prestazioni

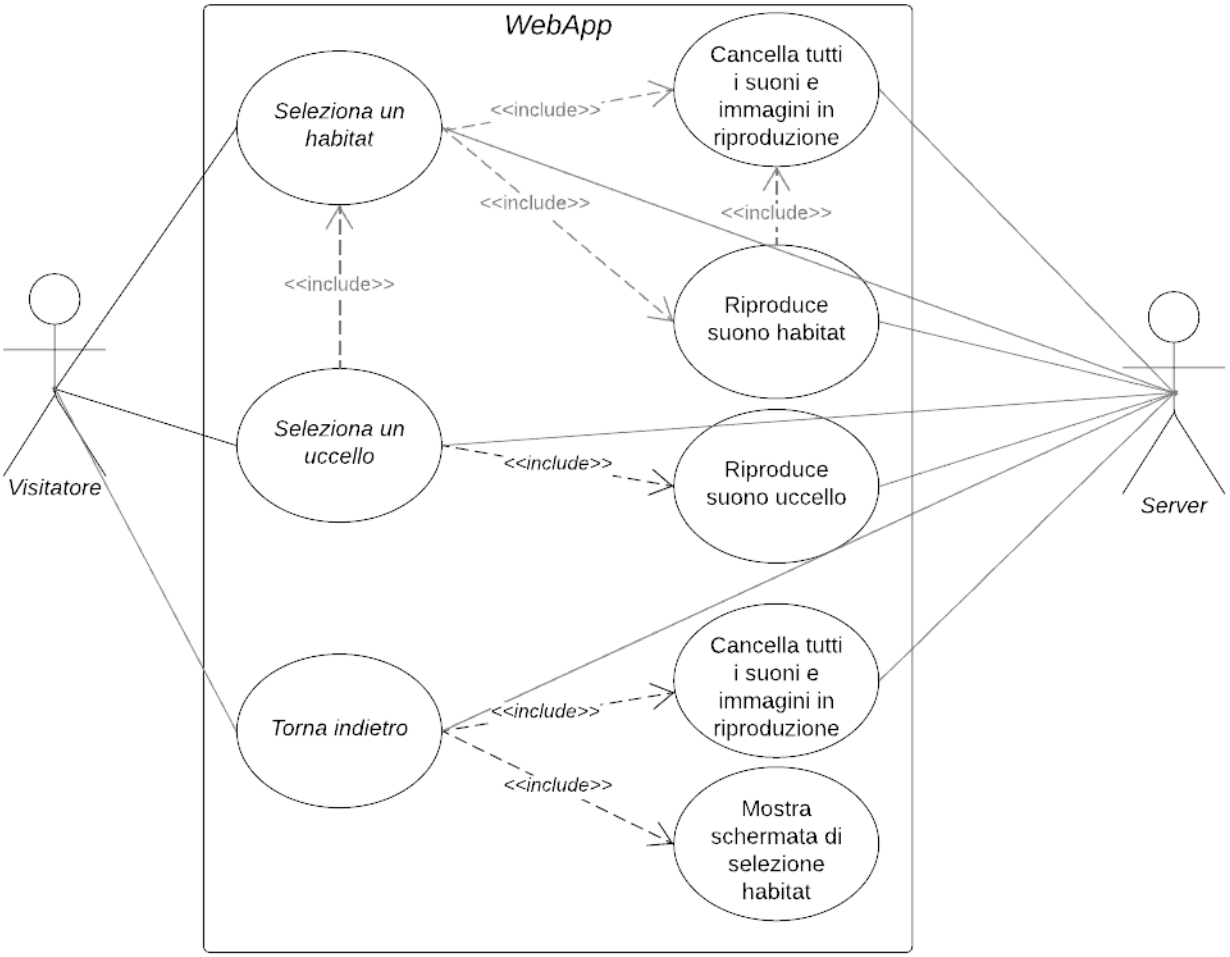
Per bilanciare scalabilità e prestazioni si sono dovuti trovare dei compromessi per quanto riguarda l'inserimento dei file audio e immagine. Come già descritto nel capitolo [3.3.], infatti, i file audio, che si ricorda possono essere solo di tipo .wav, e i file immagine, solo di tipo .png , devono seguire uno standard nomenclativo una volta aggiunti nella cartella "data" nella directory del server Processing. Questo garantisce però, a fronte di una iniziale limitazione nella libertà di un possibile inserimento futuro, una notevole rapidità e assoluta correttezza nel reperimento del file multimediale da parte del programma. Date però queste limitazioni nomenclative e di tipo per quanto riguarda i file, non ci si deve invece preoccupare della dimensione delle immagini. Sempre con l'obiettivo di garantire l'aggiornabilità futura, il codice procede già di per sé al dimensionamento, senza effettuare un rescaling della risoluzione, ad una dimensione di 1920x1080p. Se l'immagine che si vuole caricare ha una risoluzione troppo distante dalla dimensione sopra specificata e risulta troppo sgranata una volta riprodotta, sarà compito del programmatore competente scalarla ad una risoluzione più consona prima di inserirla in "data".

Inoltre è da considerare anche l'importazione della classe Map e non HashMap per garantire una maggiore flessibilità del codice. Map è un'interfaccia che definisce un contratto per la gestione delle mappe chiave-valore, mentre HashMap è una delle implementazioni concrete di tale interfaccia. Ad esempio, se in futuro si desiderasse cambiare l'implementazione da HashMap a un'altra implementazione di Map, come LinkedHashMap o TreeMap, si dovrebbe solo modificare la parte in cui si crea l'oggetto Map senza dover cambiare il resto del codice. In sintesi, l'utilizzo dell'interfaccia Map è una buona pratica poiché rende il codice più flessibile e manutenibile, anche se l'uso diretto di HashMap può essere appropriato in situazioni in cui si sa con certezza che si desiderano esclusivamente le caratteristiche specifiche di questa implementazione.

4.3.6. Diagrammi UML

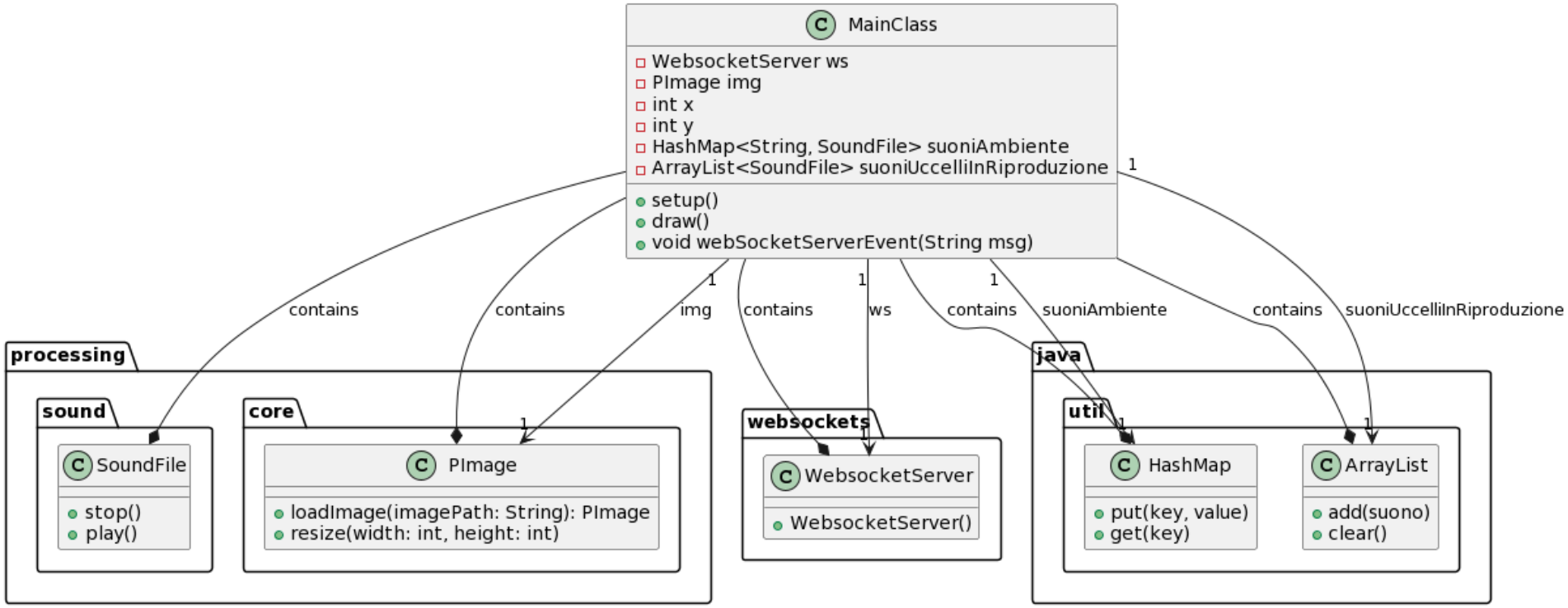
Questo paragrafo è dedicato all'illustrazione visiva della progettazione del sistema mediante l'utilizzo di diagrammi UML (Unified Modeling Language). Gli UML sono strumenti di modellazione che consentono di rappresentare in modo chiaro e preciso la struttura e le interazioni all'interno del nostro web server. Questi diagrammi forniscono una visione schematica dei casi d'uso, delle classi e delle sequenze di azioni all'interno del sistema, offrendo così un valido supporto nella comprensione e nella comunicazione della progettazione del software. Attraverso questa sezione, verranno esplorati in dettaglio i diversi tipi di diagrammi UML utilizzati, descrivendo il loro ruolo nella definizione dell'architettura, nell'analisi dei requisiti, e nel fornire una base solida per il successivo processo di sviluppo e implementazione del web server.

1. Diagramma dei casi d'uso
   Questo diagramma fornisce una visione ad alto livello delle interazioni tra il sistema e gli attori esterni. Può aiutare a definire i requisiti funzionali del sistema e a identificare i principali casi d'uso.



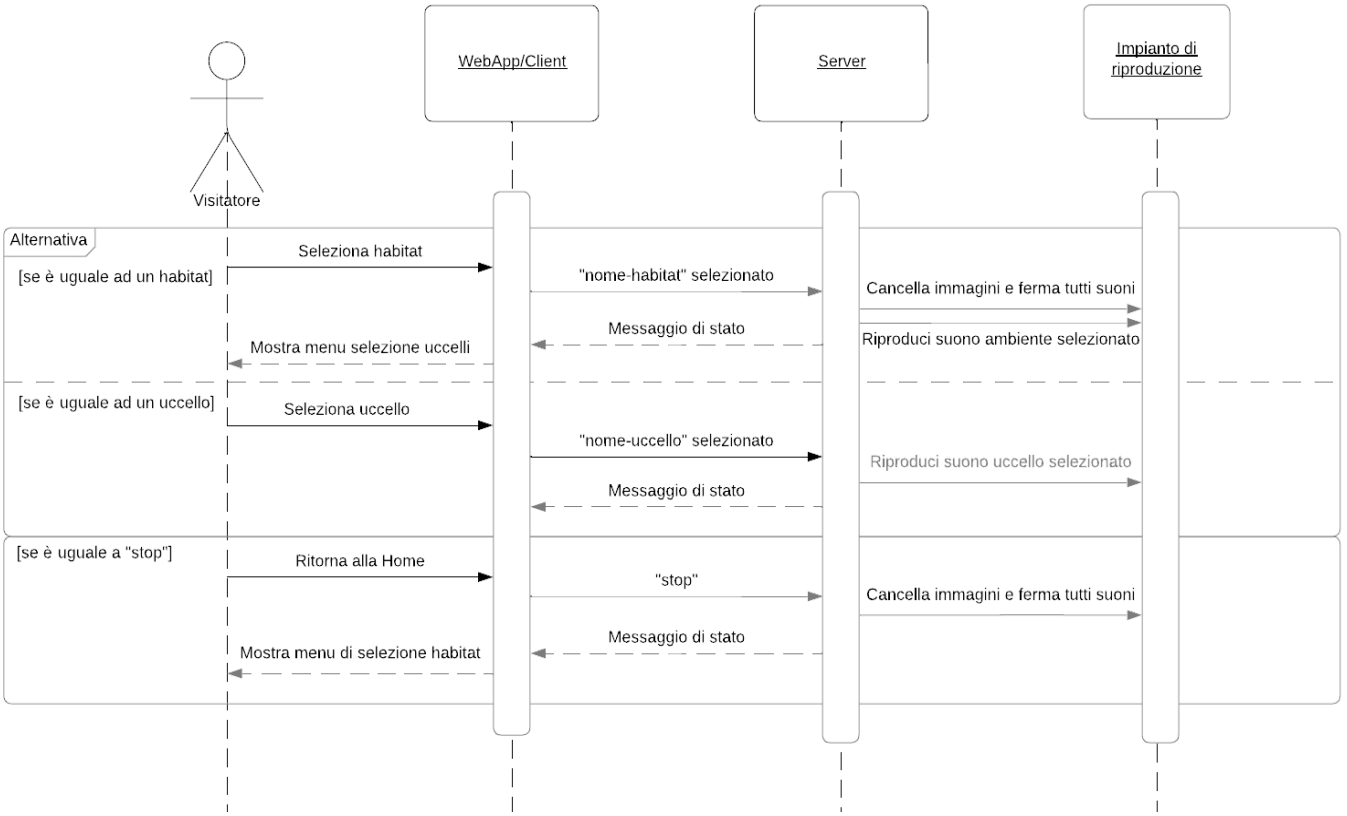
2. Diagramma delle classi

Dopo il diagramma dei casi d'uso, il diagramma delle classi è spesso presentato. Questo diagramma offre una panoramica delle classi nel sistema, delle loro relazioni e degli attributi e metodi associati. Aiuta a definire la struttura statica del sistema.


MainClass
WebsocketServer ws
PImage img
int x
int y
HashMap<String, SoundFile> suoniAmbiente
ArrayList<SoundFile> suoniUccelliInRiproduzione
setup()
draw()
void webSocketServerEvent(String msg)
contains
contains
img
contains
ws
contains
suoniAmbiente
contains
suoniUccelliInRiproduzione
1
processing
sound
SoundFile
stop()
play()
core
PImage
loadImage(imagePath: String): PImage
resize(width: int, height: int)
websockets
WebsocketServer
WebsocketServer()
java
util
HashMap
put(key, value)
get(key)
ArrayList
add(suono)
clear()

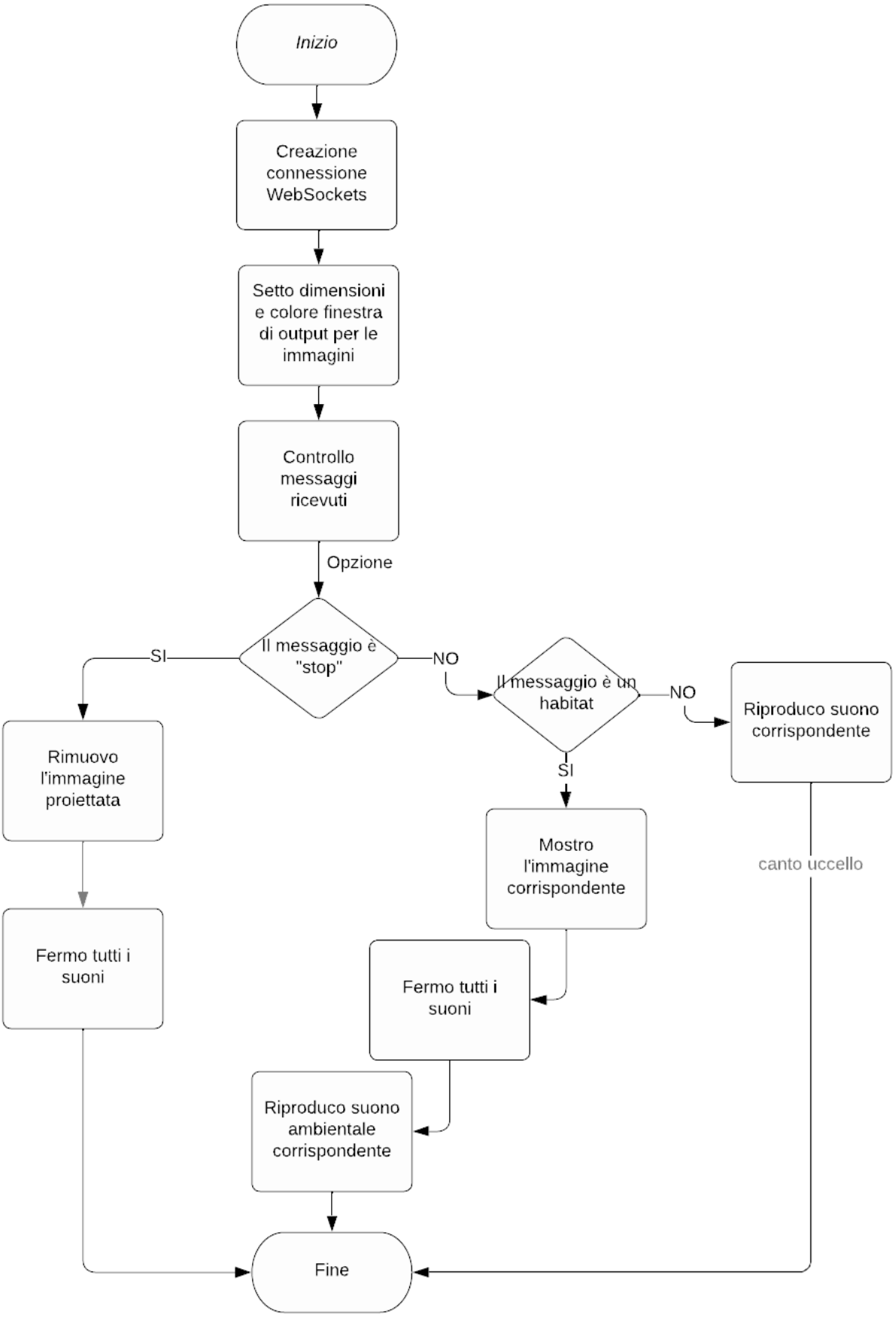
3. Diagramma delle sequenze

Il diagramma delle sequenze rappresenta l'interazione tra gli oggetti nel sistema nel corso del tempo. Questo può essere utile per comprendere il flusso delle operazioni e le sequenze temporali durante l'esecuzione di un'azione.



4. Diagramma di flusso (flowchart)

Il diagramma di flusso rappresenta visivamente la sequenza logica delle operazioni all'interno di un sistema o di un processo. Questa rappresentazione grafica offre una chiara visualizzazione del flusso di controllo, delle decisioni e delle azioni che si verificano durante l'esecuzione. Nel contesto del progetto di sviluppo del web server per il Museo di Galzignano, il diagramma di flusso sarà uno strumento prezioso per illustrare in modo intuitivo e comprensibile il percorso delle operazioni all'interno dell'implementazione del server. Attraverso questa rappresentazione, sarà possibile esplorare il flusso di esecuzione del codice, evidenziare le scelte decisionali cruciali e fornire una panoramica chiara delle interazioni chiave tra i componenti del sistema.


Inizio
Creazione connessione WebSockets
Setto dimensioni e colore finestra di output per le immagini
Controllo messaggi ricevuti
Opzione
Il messaggio è "stop"
SI
NO
Rimuovo l'immagine proiettata
Fermo tutti i suoni
Il messaggio è un habitat
NO
SI
Riproduco suono corrispondente
canto uccello
Mostro l'immagine corrispondente
Fermo tutti i suoni
Riproduco suono ambientale corrispondente
Fine

4.5. SVILUPPO E IMPLEMENTAZIONE DEL WEB SERVER

Nel presente capitolo, esamineremo il processo di sviluppo e implementazione di un web server basato sulla piattaforma Processing. Questo server è progettato per gestire richieste WebSocket e interagire con un client remoto per la visualizzazione di immagini e la riproduzione di suoni ambientali e degli uccelli. Il server agisce come intermediario tra il client e le risorse audiovisive, consentendo un'esperienza interattiva e coinvolgente.

4.5.1. Descrizione dettagliata dell'implementazione

In Processing, il metodo '`setup()`' e il metodo '`draw()`' sono due delle funzioni principali utilizzate per creare un'applicazione grafica interattiva.
Il metodo '`setup()`' è chiamato una volta sola all'avvio dell'applicazione, serve per eseguire l'inizializzazione iniziale dell'applicazione, come la configurazione della finestra grafica, l'inizializzazione di variabili, la preparazione di risorse o il setup di connessioni di rete. È qui che bisogna impostare la dimensione della finestra, caricare immagini, inizializzare oggetti, definire costanti e altre operazioni di setup necessarie.
Il metodo '`draw()`' è chiamato in un loop continuo dopo '`setup()`'. Serve per gestire il rendering degli elementi grafici, l'animazione, l'interattività e altre attività che devono essere eseguite in modo continuo durante l'esecuzione dell'applicazione. Ogni chiamata a '`draw()`' aggiorna il contenuto visibile sulla finestra grafica e consente di creare animazioni, giochi, visualizzazioni dinamiche e altre interazioni grafiche.

Client

Di seguito viene spiegato il codice dell'applicazione web client scritta in Processing che comunica tramite WebSocket con un server remoto. Il client realizzato ha la funzione di client di prova per testare il corretto funzionamento del server. Come verrà descritto nel capitolo [4.6.] sarà la web app che fungerà da client inviando dei messaggi corrispondenti alle azioni che l'utente svolgerà nella sua interfaccia. Non è l'obiettivo di questa tesi occuparsi della realizzazione del client web finale nè dell'integrazione del server con la web app, ma per fornire una visione quanto più chiara possibile al lettore ecco una descrizione dettagliata dell'implementazione del client web di prova:

1) Importazione della libreria WebSocket:

```
import websockets.*;
```

L'applicazione utilizza la libreria WebSocket di Processing per stabilire e gestire la connessione WebSocket con un server.

2) Dichiarazione delle variabili:

```
WebsocketClient client;
String inputText="";
```

- '`client`': Viene dichiarato un oggetto `WebsocketClient` che verrà utilizzato per gestire la connessione WebSocket con il server.
- '`inputTex`': Questa variabile viene utilizzata per raccogliere l'input dell'utente. :

3) `Setup()`:

```
void setup() {
     size(200, 200);
     client = new WebsocketClient(this, "ws://localhost:8025/john");
}
```

- Viene impostata la dimensione della finestra grafica.
- Viene istanziato un oggetto `WebsocketClient` che rappresenta la connessione WebSocket.
- Viene specificato il server di destinazione tramite l'URL `ws://localhost:8025/john` dove "`/john`" è un'indicazione di percorso all'interno del server WebSocket. In termini pratici, si può utilizzare "`/john`" per creare canali WebSocket distinti all'interno dello stesso server.

4) `Draw()` :

```
void draw() {
     background(220);
     textAlign(CENTER, CENTER);
     textSize(32);
     fill(0);
```

```
    text(inputText, width / 2, height / 2);
  }
```

- Il metodo ‘`draw()`’ viene utilizzato per disegnare l'interfaccia grafica dell'applicazione. Nel centro della finestra viene visualizzato il contenuto della variabile `inputText`.

5) `KeyPressed()` :

```
void keyPressed() {
  if (key >= '0' && key <= '9') {
    inputText += key;
  } else if (key == '\n') {
    if (!inputText.isEmpty()) {
      String message = getCustomMessage(inputText);
      client.sendMessage(message);
      inputText = "";
    }
  } else if (key == 'c' || key == 'C') {
    client.sendMessage("stop");
  }
}
```

- Questa funzione viene chiamata ogni volta che viene premuto un tasto sulla tastiera.
- Se il tasto premuto è un numero (0-9), il carattere corrispondente viene aggiunto alla variabile `inputText`.
- Se il tasto premuto è "`Invio`" (`'\n'`), viene creato un messaggio utilizzando la funzione ‘`getCustomMessage(inputText)`’ e inviato al server WebSocket. Successivamente, `inputText` viene azzerato.
- Se il tasto premuto è `'c'` o `'C'`, viene inviato un messaggio `"stop"` al server.

6) Funzione `getCustomMessage()` :

```
String getCustomMessage(String input) {
  // Assegna un messaggio diverso in base all'input numerico
  switch (input) {
    case "0":
```

```
      return "Alzavola"; // Umido
    case "1":
      return "Averla-piccola"; // Prativo
    case "2":
      return "Storno"; // Antropico
    case "3":
      return "Poiana"; // Roccioso + Coltivato
    case "4":
      return "Gazza"; // Prativo + Antropico
    case "5":
      return "Ghiandaia"; // Boschivo
    case "6":
      return "Antropici";
    case "7":
      return "Boschivi";
    case "8":
      return "Prativi";
    case "9":
      return "Pianure";
    case "10":
      return "Rocciosi";
    case "11":
      return "Umidi";
    default:
      return "Non è stato selezionato nessun ambiente o
uccello";
  }
}
```

-

Questa funzione restituisce un messaggio personalizzato in base all'input numerico fornito come argomento. I numeri da 0 a 11 vengono mappati a messaggi specifici che verranno inviati al server. Si nota che il client è solo d'esempio e utile in fase di test ma tuttavia non completo in quanto dovrebbero esserci oltre ai messaggi esistenti anche quelli di tutti gli altri uccelli mancanti.

Server

Anche l'implementazione del web server è stata realizzata utilizzando il linguaggio di programmazione Processing. Il server utilizza la libreria "Websockets" per gestire le comunicazioni WebSocket con il client remoto. La principale funzione del server è ricevere messaggi dal client, interpretarli e agire di conseguenza.

Il server è in grado di gestire le seguenti tipologie di messaggi:

- `"stop"`: questo messaggio interrompe qualsiasi attività in corso, cancella eventuali immagini visualizzate e interrompe la riproduzione di suoni ambientali e degli uccelli.
- Messaggi relativi agli habitat: quando il client invia il nome di un habitat (ad esempio, `"Boschivi"` o `"Prativi"`), il server carica l'immagine corrispondente e avvia il suono ambientale specifico per quell'habitat.
- Messaggi relativi agli uccelli: se il client invia il nome di un uccello (ad esempio, `"Storno"` o `"Alzavola"`), il server carica e riproduce il suono dell'uccello selezionato.

Di seguito viene descritta l'implementazione del codice che funge da server WebSocket e gestisce la visualizzazione di immagini e la riproduzione di suoni ambientali e di uccelli in base ai messaggi ricevuti da un client WebSocket. Ecco una descrizione dettagliata dell'implementazione:

1) Librerie importate:
   - Il codice importa le librerie "`processing.sound.*`" e "`websockets.*`" necessarie per la gestione dei suoni e delle connessioni WebSocket. In Processing, alcune funzionalità di base, come l'uso di mappe (HashMap) e liste (ArrayList), sono disponibili senza la necessità di importare librerie aggiuntive. Questo è dovuto al fatto che Processing include di default molte librerie standard che semplificano lo sviluppo creativo e artistico. Pertanto, molte funzionalità comuni sono integrate senza richiedere importazioni esplicite di librerie. Le mappe e le liste sono strutture dati standard all'interno del core di Java che vengono spesso utilizzate, quindi hanno una presenza nativa in Processing come parte integrante del linguaggio di programmazione Java.

2) Variabili globali:

```
WebsocketServer ws;
PImage img;
int x, y;
HashMap<String, SoundFile> suoniAmbiente = new HashMap<String,
SoundFile>();
ArrayList<SoundFile> suoniUccelliInRiproduzione = new
ArrayList<SoundFile>();
```

   -
   Viene dichiarato un oggetto "`WebsocketServer`" chiamato "`ws`" per gestire le connessioni WebSocket.
   - Vengono dichiarate le variabili "`img`" per l'immagine relativa agli habitat da proiettare e "`x`" e "`y`" per la posizione dell'immagine visualizzata.

- Viene creata una "`HashMap`" chiamata "`suoniAmbiente`" per mappare nomi di habitat agli oggetti "`SoundFile`" che rappresentano suoni ambientali.
- Viene dichiarato un "`ArrayList`" chiamato "`suoniUccelliInRiproduzione`" per tenere traccia dei suoni degli uccelli in riproduzione.

3) `setup()` :

- Nel setup, viene inizializzato il canvas di Processing con una dimensione di 580x430 pixel.
- Viene inizializzato il server WebSocket sulla porta `8025` e con il percorso "`/john`" utilizzando "`new WebsocketServer(this, 8025, "/john")`".
- Vengono inizializzati i suoni ambientali caricandoli dai file audio corrispondenti a ciascun habitat e memorizzandoli nell'HashMap "`suoniAmbiente`":

```
String[] habitat = {"Antropici", "Boschivi", "Pianure", "Prativi", "Rocciosi",
"Umidi"};
  for (String h : habitat) {
    String soundPath = dataPath(h + ".wav"); // Costruisce il percorso al file
audio dell'habitat.
    SoundFile suonoAmbiente = new SoundFile(this, soundPath); // Crea un
oggetto SoundFile per il suono dell'habitat.
    suoniAmbiente.put(h, suonoAmbiente); // Aggiunge il suono ambiente alla
mappa "suoniAmbiente" utilizzando il nome dell'habitat come chiave.
  }
```

4) `draw()` :

- Nel metodo "`draw`", lo sfondo viene impostato su nero `background(0)`.
- Se è presente un'immagine nell'oggetto "`img`", essa viene ridimensionata in FHD e visualizzata sulla posizione specificata da "`x`" e "`y`".

```
if (img != null) {
    img.resize(1920, 1080);
    image(img, x, y); // Se c'è un'immagine, la mostro
  }
```

5) `webSocketServerEvent(String msg)` :

- Questo metodo viene chiamato ogni volta che il server WebSocket riceve un messaggio dal client.

- Se il messaggio è "stop", l'applicazione cancella il disegno, rimuove l'immagine e ferma tutti i suoni ambientali e degli uccelli in riproduzione.

```
if (msg.equalsIgnoreCase("stop")) {
    clear(); // Cancella proiezione habitat
    img = null;
    for (SoundFile suonoAmbiente : suoniAmbiente.values()) {
      suonoAmbiente.stop();
    }
    for (SoundFile suono : suoniUccelliInRiproduzione) {
      suono.stop();
    }
    suoniUccelliInRiproduzione.clear(); // Svuoto l'elenco dei suoni
degli uccelli
  }
```

- Se il messaggio corrisponde ad un habitat (presente nell' HashMap "`suoniAmbiente`"), l'applicazione proietta l'immagine dell'habitat selezionato, ferma il precedente suono ambientale, degli uccelli in riproduzione e pulisce la lista suoni, infine avvia il suono ambiente specifico per l'ultimo habitat selezionato.

```
else if (suoniAmbiente.containsKey(msg)) {
    String imagePath = dataPath(msg + ".png");
    img = loadImage(imagePath);
    x = 0;
    y = 0;

    for (SoundFile suono : suoniUccelliInRiproduzione) {
      suono.stop();
    }
    suoniUccelliInRiproduzione.clear();

    for (SoundFile suono : suoniAmbiente.values()) {
      suono.stop();
    }

    suoniAmbiente.get(msg).play(); // Avvio il suono ambiente specifico
per l'habitat selezionato
  }
```

- Se il messaggio non corrisponde a un habitat, viene caricato e riprodotto il suono degli uccelli, e viene mantenuta una lista dei suoni degli uccelli in riproduzione.

```
else {
    String soundPath = dataPath(msg + ".wav");
    SoundFile suono = new SoundFile(this, soundPath);
    suono.play();
    suoniUccelliInRiproduzione.add(suono);
  }
```

In sintesi, questo codice crea un server WebSocket in Processing che permette a un client di inviare messaggi per cambiare l'immagine visualizzata e riprodurre suoni ambientali o suoni degli uccelli in base agli habitat selezionati. L'applicazione è progettata per rispondere in tempo reale ai messaggi inviati dal client attraverso una connessione WebSocket, fornendo una rappresentazione multimediale di diversi habitat e suoni della natura. Si veda [5. APPENDICE] per il codice completo.

4.5.2. Struttura del codice

Il codice del server è organizzato in modo chiaro e modulare. È suddiviso in funzioni e metodi per gestire diverse operazioni, tra cui la gestione delle comunicazioni WebSocket, il caricamento di immagini, la riproduzione di suoni e la gestione delle richieste del client. Il codice è strutturato per garantire la leggibilità e la manutenibilità.

4.5.3. Configurazione e personalizzazione

Il server è configurato per ascoltare le connessioni WebSocket sulla porta `8025` e sul percorso `"/john"`. Questa configurazione è adattabile alle esigenze specifiche del progetto. Inoltre, il server è personalizzabile in termini di risorse audiovisive, consentendo l'aggiunta di nuove immagini e suoni ambientali per habitat e uccelli.

4.5.4. Risoluzione di problemi e debug

Nel processo di sviluppo del server, sono state affrontate diverse sfide relative alla gestione delle comunicazioni WebSocket, al caricamento delle risorse audiovisive e al controllo della riproduzione

dei suoni. Durante il debug, sono stati utilizzati strumenti di monitoraggio delle comunicazioni per individuare e risolvere eventuali errori. In particolare sono stati usati:

- Console del server Processing: per stampare messaggi di log personalizzati e informazioni di debug per tenere traccia delle attività e degli eventi
- Simulazione di errori: sono stati deliberatamente introdotti errori nel codice durante lo sviluppo per testare la gestione degli errori. Ad esempio, interrompere temporaneamente la connessione di rete o simulare un comportamento anomalo per vedere come il sistema reagiva

Nel codice del client fornito, non sono state previste esplicite gestioni degli errori per situazioni come la pressione di un tasto che non corrisponde a nessun messaggio del client. Questo perché il client è stato creato come prototipo o client di prova, e nell'applicazione web finale, ogni tasto dell'interfaccia utente dovrebbe essere mappato a una funzione specifica. Questo azzera completamente la possibilità di errori dovuti a input inaspettati da parte dell'utente.
Il controllo degli input errati o inattesi dovrebbe essere gestito principalmente dal front-end dell'applicazione, assicurandosi che l'interfaccia utente consenta solo interazioni valide.
Attualmente, in caso di input non valido, il server manda un messaggio di errore `NullPointerException` al seguito del quale bisogna riavviare la connessione col client per inoltrare nuovi messaggi. Di fatto oltre a questo tipo di errore, in fase di test non si sono verificati errori degni di nota o che richiedano una gestione specifica.

In ultima analisi si parlerà della documentazione Processing usata per implementare le funzionalità necessarie e che è stata utile nel risolvere alcuni problemi che si sono verificati durante lo sviluppo del server e del client di prova.

- PImage : una libreria standard si Processing per la gestione di immagini 2D e 3D, che fornisce strumenti essenziali per la memorizzazione e la manipolazione di tali risorse visive. La comprensione approfondita delle variabili di tipo ‘PImage’ è stata acquisita attraverso la consultazione della documentazione ufficiale. Queste variabili sono state impiegate per la proiezione delle maschere territoriali nel contesto della realizzazione del plastico 3D del museo.
- HashMap : una libreria standard di Java che fornisce funzionalità per gestire mappe chiave-valore (noto anche come dizionario) all'interno del codice. È stato utile verificarne il funzionamento per implementare la HashMap "`suoni Ambiente`" per la gestione dei file audio dei sei habitat.

- ArrayList : la classe ArrayList fa parte del core di Java e fornisce una struttura dati dinamica che può essere utilizzata per gestire elenchi di oggetti (in questo caso suoni "`SoundFile`"), non è necessario importarla ed è stata utilizzata per operare con la lista "`suoniUccelliInRiproduzione`", struttura dati essenziale per il funzionamento del server.
- SoundFile : utilizzata per caricare, riprodurre e manipolare file audio all'interno di un programma creato con Processing, si possono usare sia stop() che pause() per fermare gli audio in riproduzione.
- dataPath : il metodo `dataPath()` in Processing è un metodo della classe PApplet. PApplet è una classe base di Processing ed è la classe principale che rappresenta l'applicazione stessa (ovvero significa che quando si scrive un programma in Processing, si sta creando un oggetto di tipo PApplet che rappresenta lo sketch o applicazione). Il metodo `dataPath()` viene utilizzato per ottenere il percorso completo dei file audio e immagine all'interno della cartella "data" del progetto.

4.6. INTEGRAZIONE CON LA WEB APP

La web app Java funge da client nell'architettura client-server, interagendo con l'utente attraverso un'interfaccia grafica. Quando l'utente seleziona un'opzione, come un pulsante per avviare la riproduzione di una risorsa audiovisiva, la webapp genera un messaggio rappresentativo dell'azione compiuta dall'utente.

Questo messaggio, usato dal server per identificare il file o i file da riprodurre, viene quindi inviato al server attraverso il protocollo WebSocket, che consente una comunicazione bidirezionale e in tempo reale. Il server, basato su Processing, è configurato per ricevere e interpretare questi messaggi in arrivo attraverso il protocollo WebSocket.
Una volta ricevuto il messaggio, il server interpreta se si tratta di un file audio o immagine e esegue l'azione corrispondente. La webapp è responsabile di gestire gli eventi utente, creare i messaggi corretti e inviarli al server utilizzando il protocollo WebSocket.
Dopo l'esecuzione dell'azione richiesta, il server invia un messaggio di conferma o di stato all'applicazione client, informando la webapp sull'esito dell'operazione. La webapp può quindi aggiornare l'interfaccia utente di conseguenza, fornendo feedback all'utente sull'avanzamento dell'azione e riflettendo lo stato attuale della risorsa audiovisiva. Questo ciclo di interazione client-server, basato sul protocollo WebSocket, consente una gestione fluida delle richieste

dell'utente e delle operazioni sul server senza dover descrivere il funzionamento interno del server, che è già noto.

4.7. TESTING E VALUTAZIONE

Questo capitolo rappresenta una fase cruciale del processo di sviluppo del sistema web. Questa sezione si concentra sulla progettazione e l'esecuzione di test per verificare l'affidabilità, le prestazioni e la funzionalità del sistema. Inoltre, verranno valutati i risultati dei test e discussi i miglioramenti e le sfide emerse durante il processo.
Gli obiettivi dei test includono la verifica della stabilità del sistema web server e la sua capacità di gestire un carico di lavoro sostenuto nel tempo, come diversi messaggi in successione rapida. Si noti che non è stato necessario monitorare la capacità del server di gestire picchi di traffico improvvisi in quanto questo può, per ragioni strutturali, essere utilizzato da un solo utente alla volta. Sono stati sviluppati test per garantire che il server risponda correttamente alle diverse categorie di messaggi inviati dal client, inclusi messaggi di controllo, richieste di caricamento di immagini e richieste di riproduzione audio.
I test di prestazioni sono stati condotti per misurare il tempo di risposta del server, tra l'arrivo del messaggio e l'output dei risultati, e la latenza di comunicazione tra client e server. Questi risultati non tengono conto del tempo (di circa 2000 ms) necessario per avviare la connessione col server Jetty al primo avvio del programma (si veda capitolo [4.3.2.]), ma sono stati fatti a connessione già avviata.
Modificando il codice del client e del server come segue, è stato misurato il delay tra il momento di invio di un messaggio (`"ciao"`) da parte del client e il momento della ricezione del messaggio da parte del server :

```
// Client

import websockets.*;

WebsocketClient client;

void setup() {
  // Crea un'istanza del client WebSocket e stabilisce la connessione
con il server
  client = new WebsocketClient(this, "ws://localhost:8025/john");

  // Invia il messaggio di ping al server al momento dell'avvio
```

```
  sendPingMessage();
}
void draw() {
}

// Funzione per inviare un messaggio di ping al server
void sendPingMessage() {
  // Registra il tempo attuale in nanosecondi al momento dell'invio del
messaggio
  long sendTime = System.nanoTime();

  client.sendMessage("ciao");

  // Stampa il tempo di invio in nanosecondi sulla console
  println("Inviato alle: " + sendTime + " ns");
}
```

```
// Server

import websockets.*;

WebsocketServer ws;

void setup() {
  ws = new WebsocketServer(this, 8025, "/john");
}

void draw() {
}

void webSocketServerEvent(String msg) {
  // Divide il messaggio in tipo e contenuto del messaggio

    if (msg.equals("ciao")) {
      // Stampa l'orario in cui il server ha ricevuto il messaggio
"ciao" dal client
      println("Ricevuto alle: " + System.nanoTime() + " ns");
    }
}
```

Dove `System.nanoTime()` restituisce il tempo corrente in nanosecondi. Questa funzione è spesso utilizzata per misurare intervalli di tempo molto brevi con una precisione molto alta. Tuttavia, va

notato che la precisione effettiva può variare a seconda del sistema operativo e dell'hardware. È importante notare che `System.nanoTime()` non restituisce il tempo assoluto, ma piuttosto il tempo trascorso dal momento in cui il sistema è stato avviato. Pertanto, può essere utilizzato per misurare intervalli di tempo relativi, ad esempio, il tempo trascorso tra due eventi, come in questo caso.

In console è stampato il delay tra l'invio del messaggio da parte del client e la ricezione da parte del server:

```
Inviato alle: 62146081173887 ns
```

```
Ricevuto alle: 62146083469960 ns
```

La differenza è di 2296073 ns che corrispondono a circa 2,3 millisecondi**.** I risultati possono variare anche di un 50% ma il valore di `System.nanoTime()`, come detto prima, è dipendente da diversi fattori come precisione dell'orologio di sistema e carico di lavoro del sistema. Inoltre il tempo totale trascorso tra l'invio di un messaggio da parte del client e la ricezione di una risposta da parte del server può essere influenzato da fattori di rete come la latenza, la larghezza di banda e l'overload di rete. Se la rete è congestionata o la latenza è elevata, potrebbe essere necessario più tempo per inviare e ricevere i messaggi, influenzando il tempo totale misurato dal codice.

Per calcolare invece il tempo di elaborazione del server, misurato come il tempo dall'arrivo del messaggio all'output del contenuto audiovisivo corrispondente, è stato modificato il codice del server Processing originale (si veda [5. APPENDICE]) come segue:

```
import processing.sound.*;
import websockets.*;

// Codice come originale

long startTime; // Tempo di inizio del caricamento
long responseTime; // Tempo di risposta
```

```
// Codice come originale
```

```
void webSocketServerEvent(String msg) {
  startTime = millis(); // Registra il tempo all'inizio della funzione
  println("Messaggio ricevuto dal client: " + msg);
  // Codice come originale
  } else if (suoniAmbiente.containsKey(msg)) {
    // Codice come originale
    responseTime = millis() - startTime; // Calcola il tempo di
risposta
    println("Tempo di risposta totale per il messaggio " + msg + ": " +
responseTime + " ms");
  } else {
    // Codice come originale
    responseTime = millis() - startTime; // Calcola il tempo di
risposta
    println("Tempo di risposta totale per il messaggio " + msg + ": " +
responseTime + " ms");
  }
}
```

Introducendo una variabile "`startTime`" è stato possibile registrare il tempo di ricevimento del messaggio e aggiornando la variabile "`responseTime`" viene calcolata la differenza tra il tempo di arrivo del messaggio e il momento in cui viene restituito in output il file audio e/o immagine corrispondente. Con questo procedimento si può quindi calcolare la rapidità di elaborazione del server dei messaggi, che sommando al delay misurato al punto precedente può dare una stima dei tempi totali di esecuzione del programma.
I risultati ottenuti sono i seguenti:

```
Messaggio ricevuto dal client: Rocciosi
Tempo di risposta totale per il messaggio Rocciosi: 444 ms
Messaggio ricevuto dal client: Averla-piccola
Tempo di risposta totale per il messaggio Averla-piccola: 22 ms
Messaggio ricevuto dal client: Storno
Tempo di risposta totale per il messaggio Storno: 5 ms
Messaggio ricevuto dal client: Poiana
Tempo di risposta totale per il messaggio Poiana: 10 ms
Messaggio ricevuto dal client: Gazza
Tempo di risposta totale per il messaggio Gazza: 3 ms
Messaggio ricevuto dal client: Pianure
```

```
Tempo di risposta totale per il messaggio Pianure: 231 ms
Messaggio ricevuto dal client: Averla-piccola
Tempo di risposta totale per il messaggio Averla-piccola: 1 ms
Messaggio ricevuto dal client: Storno
Tempo di risposta totale per il messaggio Storno: 0 ms
Messaggio ricevuto dal client: Poiana
Tempo di risposta totale per il messaggio Poiana: 2 ms
Messaggio ricevuto dal client: Gazza
Tempo di risposta totale per il messaggio Gazza: 1 ms
Messaggio ricevuto dal client: Ghiandaia
Tempo di risposta totale per il messaggio Ghiandaia: 4 ms
Messaggio ricevuto dal client: Prativi
Tempo di risposta totale per il messaggio Prativi: 151 ms
Messaggio ricevuto dal client: stop
```

Si evidenzia una differenza nei tempi di elaborazione tra i messaggi ambientali e quelli degli uccelli. Questo perché, oltre alle operazioni di caricamento e riproduzione del file audio associato, i messaggi ambientali richiedono anche il caricamento e la proiezione dell'immagine corrispondente.

4.7.2. Ambiente di test

Per condurre i test, è stato allestito un ambiente di test dedicato. Sia il server che il client erano eseguiti su sistemi locali, connessi tramite una rete locale. La configurazione dell'ambiente di test ha permesso di simulare scenari realistici di interazione tra client e server. Il codice è stato eseguito sul seguente hardware e sistemi operativi:

1) MacOS Monterey v.12.7.1 (MacBook Air 2015 : i5 dual-core 2.7GHz, 8GB ddr3)
2) Microsoft Windows 11 23H3 (Ryzen 7 5800X3D 8c/16t 4.5GHz, 32GB ddr4)

4.7.3. Test di affidabilità

I test di affidabilità hanno incluso l'esecuzione continua del server per un periodo prolungato per verificare la sua stabilità. Il server una volta avviato rimane in esecuzione e continua ad eseguire le richieste inviate dal client fino a quando non viene esplicitamente chiuso. Non sono stati condotti test di resistenza per determinare il limite massimo di connessioni WebSocket supportate dal server per verificare problemi di sovraccarico in quanto il server non è stato creato per gestire connessioni multiple. Nella realtà di interesse è presente un solo totem del museo su cui gira il server e la web app (client) che invia le richieste.

4.7.4. Test di funzionalità

I test di funzionalità hanno verificato la corretta interpretazione dei messaggi inviati dal client. Sono stati simulati scenari in cui il client inviava messaggi relativi a habitat e uccelli, e il server doveva rispondere correttamente mostrando immagini e riproducendo i suoni associati.

4.7.5. Test di prestazioni

I test di prestazioni hanno incluso la misurazione del tempo di risposta del server alle richieste del client. Sono stati analizzati i tempi di caricamento delle risorse audiovisive e la latenza di comunicazione tra le due parti. [Vedere capitolo 4.7.]

4.7.6. Valutazione dei risultati

I risultati dei test sono stati attentamente analizzati. Il server ha dimostrato un'elevata stabilità e i test di funzionalità hanno confermato che il server risponde correttamente alle richieste del client, visualizzando immagini e riproducendo i suoni previsti. I test di prestazioni hanno mostrato tempi di risposta accettabili e una comunicazione efficiente tra il client e il server.

4.7.7. Sfide e miglioramenti

Durante il processo di testing, alcune sfide sono emerse, tra cui la gestione delle risorse audiovisive e la sincronizzazione tra immagini e suoni. Alcuni miglioramenti futuri includono l'ottimizzazione del codice per prestazioni ancora migliori e l'aggiunta di ulteriori risorse audiovisive per una maggiore varietà. In particolare si potrebbe semplificare il processo di inserimento di nuovi habitat e uccelli. Per aggiungere un nuovo habitat ora bisogna aggiungere il file “Nuovohabitat.wav” e Nuovohabitat.png nella cartella ‘data’, aggiungere il messaggio “Nuovohabitat” nel client e anche alla Hashmap del server. Una miglioria futura, anche se tutt’ora non necessaria, potrebbe essere quella di poter aggiungere un file audio o immagine senza preoccuparsi dell’estensione, in quanto il server stesso cercherà se esiste quel file indipendentemente dall’estensione che ha e in base all’estensione che trova decidere se è da proiettare o da riprodurre negli speaker. Come anche non rendere obbligatoria la nomenclatura di file multimediali e messaggi client, anche se questo causerebbe un notevole aumento dei tempi di esecuzione del server, incrementando il tempo di ricerca, per trovare il file giusto.

In conclusione, si è confermata la solidità del sistema web server sviluppato. I test hanno dimostrato che il server è in grado di gestire un carico di lavoro significativo e rispondere in modo adeguato alle richieste del client. Gli esiti positivi dei test sottolineano l'efficacia del sistema e forniscono una base solida per ulteriori sviluppi e implementazioni. Tuttavia, il processo di sviluppo ha anche rivelato alcune sfide, come la gestione delle risorse audiovisive, che potrebbero richiedere ulteriori raffinamenti. Nel capitolo successivo, esamineremo l'integrazione del sistema con tecnologie aggiuntive, come l'uso di sensori per l'interazione fisica, e discuteremo le possibilità di futuri sviluppi per rendere il sistema ancora più robusto e interattivo. Inoltre, verranno valutate le implicazioni pratiche dell'implementazione del sistema, comprese le potenziali applicazioni in contesti reali come musei, parchi naturali e centri educativi.

4.8. CONCLUSIONI E FUTURE DIREZIONI

Il presente capitolo rappresenta il culmine del percorso di ricerca e sviluppo condotto nell'ambito della tesi di laurea in ingegneria informatica. In questa sezione, verranno esposte le conclusioni derivanti dall'intera esperienza, compreso il processo di progettazione e implementazione del sistema web server basato su Processing. Inoltre, saranno delineate le potenziali direzioni future della ricerca, offrendo spunti e obiettivi per ulteriori sviluppi nell'ambito del progetto.

4.8.1. Riassunto dei risultati ottenuti

I risultati ottenuti attraverso il percorso di sviluppo del sistema web server e il conseguente processo di testing e valutazione sono stati, nel complesso, altamente positivi. Il server ha dimostrato di essere stabile e affidabile, in grado di gestire un carico di lavoro notevole e rispondere in modo adeguato alle richieste del client. La corretta interpretazione dei messaggi inviati dal client, la visualizzazione di immagini e la riproduzione di suoni ambientali e degli uccelli hanno confermato l'efficacia del sistema nel raggiungere gli obiettivi prefissati.

In particolare, il sistema si è dimostrato adatto per scenari che coinvolgono la presentazione audiovisiva di habitat naturali e l'interazione con il pubblico. Questo aspetto potrebbe avere applicazioni significative in contesti come parchi naturali, centri educativi e musei, offrendo esperienze coinvolgenti e educative agli utenti.

4.8.2. Contributi della tesi

La tesi ha apportato diversi contributi significativi nell'ambito dell'ingegneria informatica:

- Sviluppo del sistema web server: l'implementazione del web server basato su Processing costituisce un contributo chiave. Il codice sorgente, organizzato in modo modulare e ben documentato, offre una base per ulteriori sviluppi e applicazioni in cui sia necessaria l'interazione tra contenuti audiovisivi e utenti.

- Test e valutazione approfonditi: la progettazione e l'esecuzione dei test hanno consentito di dimostrare l'affidabilità del sistema e di identificare le sue prestazioni e le potenziali aree di miglioramento.

- Applicazioni potenziali: i risultati della tesi suggeriscono l'applicabilità del sistema in ambienti educativi e culturali, aprendo nuove prospettive per la divulgazione scientifica e l'interazione multimediale.

Nel corso dello sviluppo del progetto di tesi, sono emerse una serie di soft skill fondamentali che hanno contribuito in modo significativo al successo del lavoro. Queste competenze includono abilità di problem-solving, capacità di progettazione, apprendimento autonomo e gestione del tempo.

- Abilità di problem-solving: durante lo sviluppo del progetto, sono state affrontate diverse sfide tecniche, come la gestione efficiente di grandi quantità di dati audio e immagini, l'ottimizzazione delle prestazioni del server e la gestione degli errori di comunicazione. È stata sviluppata la capacità di analizzare tali problemi in modo sistematico e di trovare soluzioni efficaci.

- Capacità di progettazione: l'architettura del server e dell'applicazione web è stata progettata in modo da garantire la scalabilità, la sicurezza e la manutenibilità del sistema. Si è tradotto requisiti complessi in una progettazione chiara e modulare.

- Abilità di autodidattica: sono state acquisite nuove conoscenze tecniche per completare il progetto con successo. È stata dimostrata la capacità di acquisire conoscenze tecniche in modo autonomo e di applicarle al contesto del progetto.

- Capacità di gestione del tempo: il progetto è stato pianificato, stabilendo scadenze e obiettivi intermedi, dimostrando la capacità di gestire efficacemente le risorse temporali e di mantenere il progetto in linea con il programma stabilito.

4.8.3. Possibili sviluppi futuri e ricerche correlate

L'ambito di ricerca aperto dalla tesi offre molte opportunità per futuri sviluppi. Alcune delle direzioni possibili includono:

- Integrazione di sensori fisici: l'aggiunta di sensori ambientali o di posizione potrebbe consentire un'interazione fisica più avanzata con il sistema, ad esempio, consentendo agli utenti di influenzare l'esperienza audiovisiva attraverso il movimento o il rilevamento di stimoli esterni.

- Personalizzazione dell'esperienza: sviluppare un sistema di personalizzazione per consentire agli utenti di selezionare habitat, specie di uccelli o percorsi interattivi, ampliando così le possibilità di coinvolgimento.

- Esplorazione di tecnologie web avanzate: l'adozione di tecnologie web all'avanguardia, come WebAssembly o WebRTC, potrebbe migliorare ulteriormente le prestazioni e l'interattività del sistema.

- Ricerche correlate: l'approfondimento degli aspetti legati all'ambiente sonoro e alla percezione degli utenti potrebbe rappresentare un'area di ricerca correlata interessante. La valutazione delle reazioni degli utenti e delle implicazioni cognitive dell'esperienza potrebbe contribuire a una comprensione più approfondita dell'impatto del sistema. Inoltre, è opportuno valutare la possibilità di trattare tale circostanza come un esempio di gamification nell'ambito educativo, con l'intento di condurre uno studio mirato alla identificazione degli impatti positivi che potrebbe avere sul processo di apprendimento dei giovani fruitori.

In conclusione, la tesi ha aperto una serie di prospettive interessanti per la futura ricerca e sviluppo nel campo dell'interazione multimediale basata su web. Il sistema web server rappresenta una solida base per ulteriori esplorazioni e implementazioni, con il potenziale per offrire esperienze coinvolgenti e educative in contesti diversi. La ricerca continuerà a essere una fonte di ispirazione per l'innovazione e l'avanzamento nella scienza dell'informatica e nelle applicazioni interattive.

5. APPENDICE

```
// Server

import processing.sound.*;
import websockets.*;

WebsocketServer ws;
PImage img;
int x, y;
HashMap<String, SoundFile> suoniAmbiente = new HashMap<String,
SoundFile>(); // Mappa per i suoni ambientali
ArrayList<SoundFile> suoniUccelliInRiproduzione = new
ArrayList<SoundFile>(); // Elenco dei suoni degli uccelli in
riproduzione

void setup() {
  size(1920, 1080);
  ws = new WebsocketServer(this, 8025, "/john"); // Inizializzo il
server WebSocket
  x = 0;
  y = 0;

  // Inizializzo i suoni ambientali per gli habitat
  String[] habitat = {"Antropici", "Boschivi", "Pianure", "Prativi",
"Rocciosi", "Umidi"};
  for (String h : habitat) {
    String soundPath = dataPath(h + ".wav"); // Percorso al file audio
dell'habitat
    SoundFile suonoAmbiente = new SoundFile(this, soundPath);
    suoniAmbiente.put(h, suonoAmbiente); // Aggiunge il suono ambiente
alla mappa
  }
}

void draw() {
  background(0);
  if (img != null) {
    img.resize(1920, 1080);  // Ridimensiona l'immagine alle dimensioni
del proiettore
    image(img, x, y); // Se c'è un'immagine, la mostro
  }
}
```

```
void webSocketServerEvent(String msg) {
  println("Messaggio ricevuto dal client: " + msg); // Stampa il
messaggio ricevuto dal client
  if (msg.equalsIgnoreCase("stop")) {
    // Se il messaggio è "stop", si cancella il disegno, si rimuove
l'immagine e si fermano tutti i suoni
    // ambientali e degli uccelli in riproduzione
    img = null;
    for (SoundFile suonoAmbiente : suoniAmbiente.values()) {
      suonoAmbiente.stop();
    }
    for (SoundFile suono : suoniUccelliInRiproduzione) {
      suono.stop();
    }
    suoniUccelliInRiproduzione.clear(); // Svuoto l'elenco dei suoni
degli uccelli
  } else if (suoniAmbiente.containsKey(msg)) {
    // Se il messaggio corrisponde a un habitat, si carica e mostra
l'immagine corrispondente,
    // si fermano tutti i suoni degli uccelli e si avvia il suono
ambiente specifico
    String imagePath = dataPath(msg + ".png");
    img = loadImage(imagePath);
    x = 0;
    y = 0;

    for (SoundFile suono : suoniUccelliInRiproduzione) {
      suono.stop();
    }
    suoniUccelliInRiproduzione.clear();

    for (SoundFile suono : suoniAmbiente.values()) {
      suono.stop();
    }
    suoniAmbiente.get(msg).play(); // Avvio il suono ambiente specifico
per l'habitat selezionato

  } else {
    // Altrimenti, si carica e si riproduce il suono degli uccelli e si
tiene traccia dei suoni in riproduzione
    String soundPath = dataPath(msg + ".wav");
```

```
    SoundFile suono = new SoundFile(this, soundPath);
    suono.play();
    suoniUccelliInRiproduzione.add(suono);
  }
}
```

```
// Client

import websockets.*;

WebsocketClient client;
String inputText="";

void setup() {
  size(200, 200);
  client = new WebsocketClient(this, "ws://localhost:8025/john");
}

void draw() {
  background(220);
  textAlign(CENTER, CENTER);
  textSize(32);
  fill(0);
  text(inputText, width / 2, height / 2);
}

void keyPressed() {
  if (key >= '0' && key <= '9') {
    // Se il tasto premuto è un numero, aggiungilo all'inputText
    inputText += key;
  } else if (key == '\n') {
    // Se il tasto premuto è "Invio", invia il messaggio corrispondente
al server
    if (!inputText.isEmpty()) {
      String message = getCustomMessage(inputText);
      client.sendMessage(message);
      inputText = ""; // Reset dell'inputText
    }
  } else if (key == 'c' || key == 'C') {
    client.sendMessage("stop");
  }
}
```

```
// Il client deve adottare uno standard per scrivere i nomi nei
messaggi che invia al server
String getCustomMessage(String input) {
  // Assegna un messaggio diverso in base all'input numerico
  switch (input) {
    case "0":
      return "Alzavola"; // Umido
    case "1":
      return "Averla-piccola"; // Prativo
    case "2":
      return "Storno"; // Antropico
    case "3":
      return "Poiana"; // Roccioso + Coltivato
    case "4":
      return "Gazza"; // Prativo + Antropico
    case "5":
      return "Ghiandaia"; // Boschivo
    case "6":
      return "Antropici";
    case "7":
      return "Boschivi";
    case "8":
      return "Prativi";
    case "9":
      return "Pianure";
    case "10":
      return "Rocciosi";
    case "11":
      return "Umidi";
    default:
      return "Non e' stato selezionato nessun ambiente o uccello";
  }
}
```

6. BIBLIOGRAFIA [#a]

1) Schäfer, R. Murray. *Il paesaggio sonoro.* Casa Ricordi, 1985.
2) Schäfer, R. Murray. *The Soundscape: Our Sonic Environment and the Tuning of the World.* Simon and Schuster, 1993.
3) Schäfer, R. Murray. *The Tuning of the World.* Knopf, 1977.
4) Krause, B. *The Great Animal Orchestra: Finding the Origins of Music in the World's Wild Places.* Back Bay Books, 2013.
5) Truax, B. *Modelli e strategie per il design acustico*. CLUEB, 2001.
6) Minidio, A. *I suoni del mondo*. Guerini Scientifica, 2005.
7) Bull, M., and Back, L. *The Auditory Culture Reader.* 2nd ed., Routledge, 2015.
8) Simmel, G. "Filosofia della denigrazione." *Sociologia*, Meltemi, 2018.
9) Bull, Michael. "Soundscapes of the Car: A Critical Study of Automobile Habitation." *Car Cultures*, Routledge, 2020.
10) Thibaud, Jean-Paul. *Ascoltare i suoni della città*. 2014.
11) Fantini, M. et al. *Petropolis*. Dipartimento formazione e apprendimento della Scuola universitaria professionale della Svizzera italiana, 2017.
12) Erkizia, Xabier, et al. *Il rumore lontano*. Dipartimento formazione e apprendimento della Scuola universitaria professionale della Svizzera italiana, 2017.
13) Malatesta, M. *L'utilizzo di soundmap come strumento di monitoraggio e di programmazione per la tutela del territorio*. Tesi di laurea specialistica, Università di Bologna, Facoltà di scienze matematiche, fisiche e naturali, A.C. 2009/2010.
14) Farina, A., Buscaino, G., Ceraulo, M., Pieretti, N., *The soundscape approach for the assessment and conservation of mediterranean landscapes: principles and case studie*s, Journal of Landscape Ecology 7(1), 2014.
15) Gramann, J., *The effect of mechanical noise and natural sound on visitor experiences in units of the national park system*, Texas A&M University, NPS Social Science Research Review 1(1), 1999.
16) Biserna, E., *Sound walks mobilità, arte, suono, spazio urbano, tesi di dottorato di ricerca*, Dottorato internazionale di studi audiovisivi: cinema, musica e comunicazione, Università Degli Studi di Udine, A.C. 2011/2012.
17) Spigariolo, M. *Paesaggi sonori della scuola, legami di ascolto tra Italia e Colombia*. Tesi di laurea Magistrale in Scienze della Formazione Primaria, Università degli Studi di Padova -

Dipartimento di Psicologia, Sociologia, Pedagogia e Psicologia Applicata, Università degli Studi di Verona - Dipartimento delle Scienze Umane, A.C. 2021/2022.

18) Hempton, G. *One Square Inch of Silence: One Man's Search for Natural Silence in a Noisy World*. Free press, 2009.
19) Pavan, G., Priori, P. *Il paesaggio sonoro del bosco vetusto di Col Nero.* Centro Interdisciplinare di Bioacustica e Ricerche Ambientali - Dipartimento di Scienze della Terra e dell'Ambiente - Università di Pavia, 2007.
20) Pavan, G., Farina, A. e Stuart H. Gage, editori. *Ecoacoustics: The Ecological Role of Sounds*. John Wiley & Sons, 2017.
21) Pijanowski, Bryan C.,Villanueva-Rivera, Luis J., Dumyahn, Sarah L., Farina, A., L. Krause, Bernie L., Napoletano, Brian M, Gage, Stuart H., Pieretti, N. *Soundscape Ecology: The Science of Sound in the Landscape. BioScience*, vol. 61, no. 3, 2011, pp. 203-216.
22) Monacchi, D. *Fragments of Extinction: An Eco-Acoustic Music Project on Primary Rainforest Biodiversity.* Me Monacchi, 2013.
23) Rodaway, P. *Sensuous Geographies: Body, Sense and Place*, 1st ed., Routledge, 1994.
24) Laghezza, E., "Il paesaggio sonoro: pensieri sul libero ascolto", in *Dada. Rivista di antropologia post-globale, n.1,* 2013
25) Gourley, D.,. *HTTP: The Definitive Guide.* O'Reilly Media, 2002.
26) Wrightson, K., *Soundscape: the journal of acoustic ecology*, Vol. 1, N. 1, 2000, pp 10-13.

6.1. SITOGRAFIA [#b]

1) Rullo, Luca. "Ticino Soundmap", 2019. http://ticinosoundmap.supsi.ch/. Accessed 2 November 2023.
2) "Parco Nazionale dello Yellowstone, USA". National Park Service, https://www.nps.gov/yell/index.html . Accessed 2 November 2023.
3) "Santuario delle balene di Silver Bank, Repubblica Dominicana". Ocean Society, https://www.oceanicsociety.org/expedition/swim-with-whales-silver-bank-dominican-republic/ . Accessed 2 November 2023.
4) "Parco Nazionale di Banff, Canada". Parks Canada, https://parks.canada.ca/pn-np/ab/banff . Accessed 2 November 2023.

5) “Riserva naturale del Deserto di Atacama, Cile”. Chile Travel, https://www.chile.travel/en/where-to-go/macrozone/north-and-the-atacama-desert/ . Accessed 2 November 2023.
6) “Progetto di reintroduzione dei licaoni, Sudafrica”. HabitatOnline, https://www.habitatonline.eu/2020/02/studiare-la-territorialita-del-licaone-africano-per-garantire-una-gestione-efficace-dei-conflitti-con-luomo/ . Accessed 2 November 2023.
7) “Fragments of Extinction”. Fragments of Extinction, https://www.fragmentsofextinction.org/ Accessed 2 November 2023.
8) “Quiet Parks International”. Quiet Parks, https://www.quietparks.org/contact-us . Accessed 2 November 2023.
9) “National Park Service, USA”. National prak Service, https://www.nps.gov/index.htm . Accessed 2 November 2023.
10) “European Soundscape Award”. European Environment Agency, https://www.eea.europa.eu/highlights/the-european-soundscape-award-2013 . Accessed 2 November 2023.
11) “Suolo e territorio in Europa”. European Environment Agency, https://www.eea.europa.eu/it/segnali/segnali-2019/articoli/suolo-e-territorio-in-europa . Accessed 2 November 2023.
12) “Progetto SABIOD”. CIBRA, http://www-9.unipv.it/cibra/cibra_sabiod.html
13) “Ecoacustica: cibra unipv impegnato in un progetto nella riserva naturale integrale di Sassofratino”. Unipv, https://news.unipv.it/?p=10675 .
14) “Ecologia del Suono. Nasce l’Archivio Italiano dei Paesaggi Sonori”. Marraiafura, https://www.marraiafura.com/ecologia-del-suono-nasce-l-archivio-italiano-dei-paesaggi-sonori/ . Accessed 2 November 2023.

15) “EEA 2014”. European Environment Agency, https://www.eea.europa.eu/publications/environmental-indicator-report-2014 . Accessed 2 November 2023.
16) “EEA 2016”. European Environment Agency, https://www.eea.europa.eu/publications/circular-economy-in-europe. Accessed 2 November 2023.
17) “Acoustic monitoring 1.” *WWF-UK*, https://www.wwf.org.uk/sites/default/files/2019-04/Acousticmonitoring-WWF-guidelines.pdf. Accessed 2 November 2023.
18) Pavan, G. “International Society of Ecoacoustics (ISE).” *Google Sites*, https://sites.google.com/site/ecoacousticssociety/. Accessed 2 November 2023.
19) “Progetto Soundscape: la voce del mare spenta dal rombo dei motori.” *Euronews.com*, 6 December 2021, https://it.euronews.com/2021/12/06/ambiente-progetto-soundscape-la-voce-del-mare-spenta-dal-rombo-dei-motori. Accessed 2 November 2023.
20) Silano, Carlotta, et al. “Ambienti in ascolto. Il suono per l'educazione ai luoghi. - Spaziomusica.” *Festival Spaziomusica*, 30 September 2021, https://www.spaziomusicaproject.com/ambienti-in-ascolto-il-suono-per-leducazione-ai-luoghi/. Accessed 2 November 2023.
21) “Soundscape and ambient noise levels of the Arctic waters around Greenland.” *nature.com*, 03 Dicembre 2021, https://www.nature.com/articles/s41598-021-02255-6. Accessed 02 Novembre 2023.
22) “Soundscapes and protected area conservation: Are noises in nature making people complacent?” *Nature Conservation*, 23 November 2021, https://natureconservation.pensoft.net/article_preview.php?id=69578&skip_redirect=1&utm

_source=researcher_app&utm_medium=referral&utm_campaign=RESR_MRKT_Researcher_inbound. Accessed 2 November 2023.

7. RINGRAZIAMENTI

Concludo con profonda riconoscenza per i momenti significativi e le persone che hanno fatto parte del mio percorso universitario e del mio sviluppo personale.
I miei anni trascorsi all'università sono stati un capitolo straordinario della mia vita, durante i quali ho avuto l'opportunità di approfondire la mia conoscenza nell'ambito dell'ingegneria informatica. Questi anni hanno rappresentato un periodo di crescita e apprendimento che ha contribuito in modo determinante alla mia formazione e al mio futuro.
Desidero ringraziare il mio relatore, il Prof. Antonio Rodà, per la sua guida e il suo mentorato, per la sua disponibilità, la sua passione per l'insegnamento e la dedizione nel guidarmi attraverso questo processo di ricerca che sono stati incredibili.
Inoltre vorrei ringraziare i miei stimati compagni di corso, con i quali ho condiviso le sfide e le gioie dell'apprendimento. Le discussioni in aula, le nottate passate a studiare, e le collaborazioni su progetti sono state esperienze che hanno contribuito a plasmare la mia comprensione del mondo dell'informatica. Il vostro sostegno e la vostra amicizia sono stati fondamentali, e porterò con me queste relazioni per tutta la vita.
Infine, non posso esimermi dal ringraziare la mia famiglia per il loro incrollabile sostegno e la loro comprensione. Le parole non possono esprimere quanto io sia grato per il loro appoggio costante e per avermi permesso di perseguire i miei sogni. La mia formazione e il mio successo sono il risultato dell'amore, del supporto e dei sacrifici che hanno fatto per me.
Questo capitolo della mia vita non sarebbe stato lo stesso senza le persone e le esperienze che ho incontrato lungo il cammino. Ogni sfida superata, ogni lezione appresa e ogni momento condiviso con le persone care hanno contribuito a farmi crescere come studente e come individuo. Guardo avanti con entusiasmo alle future sfide e opportunità che incontrerò, portando con me i valori e l'educazione che ho acquisito in questi anni trascorsi all'università. Grazie a tutti voi per aver reso possibile questa straordinaria avventura.