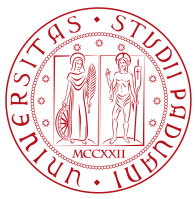

Università di Padova

Dipartimento di Ingegneria dell'Informazione

Tesi Triennale in Ingegneria Informatica

Implementazione web-based di strumenti di analisi psicoacustica

Relatore
Prof. Mauro Migliardi
Università di Padova
Dipartimento di Ingegneria dell'Informazione

Correlatore
Prof. Massimo Grassi
Università di Padova
Dipartimento di Psicologia Generale

Candidato
Mattia Toffanin

Matricola
1219608

Anno Accademico
2021-2022
Data di Laurea
16/11/2022

"ad Haka,
la mia silenziosa compagna di studi"

Abstract

Questo progetto di tesi consiste nella realizzazione e implementazione di nuove funzionalità per una piattaforma web-based per strumenti di analisi psicoacustica.
La tesi è divisa in due capitoli: nel primo si tratta la toolbox in generale esaminandone il funzionamento e i principali utilizzi, nel secondo, invece, viene descritto il lavoro effettivamente svolto durante la realizzazione di questa tesi.
La trattazione degli argomenti è accompagnata da formule, grafici, immagini e code snippet utili a una miglior comprensione. Queste si possono trovare raccolte dopo l'indice, nei relativi listati.
In breve, la toolbox offre la possibilità di stimare la propria soglia sensoriale uditiva attraverso vari test di rilevamento o discriminazione di vari stimoli. Inoltre, può essere utilizzata da ricercatori o tesisti dell'ambito per somministrare i test a soggetti terzi in modo da raccogliere dati utili agli studi in merito. La toolbox offre sei tipi di test differenti con tre diversi algoritmi per la stima della soglia.
In particolare, per la realizzazione di questo progetto di tesi, sono stati aggiunti tre nuovi tipi di test, è stata migliorata la generazione dei vari stimoli durante gli esperimenti e si è effettuato il debug della toolbox nel suo complesso.
L'obiettivo di questo progetto di tesi è rendere la toolbox pronta per il rilascio al pubblico.

Indice

Listato di immagini e grafici

Le proporzioni dei grafici non sono conformi alla realtà, ma servono a fare capire meglio le funzioni.

Listato di formule

Listato di code snippets

Introduzione

Il progetto nasce dall'idea del Professor Massimo Grassi del Dipartimento di Psicologia Generale dell'Università di Padova di creare un'applicazione che mette a disposizione vari test e esperimenti di psicoacustica ad altre persone come ricercatori o tesisti.
Il Prof. Grassi si era già occupato di creare una toolbox di questo tipo sulla piattaforma Matlab ed ottenne dei buoni risultati in termini di utilizzo. Nonostante ciò, il progetto passato aveva delle criticità e degli svantaggi evidenti. Per poter utilizzare la toolbox bisognava aver scaricato e installato Matlab, il quale è un software proprietario a pagamento che richiede abbastanza risorse, e perciò di complicato accesso. Inoltre, in seguito a vari aggiornamenti della piattaforma Matlab che non offrivano la retrocompatibilità, la toolbox è deprecata e quindi bisognava aggiornarla.
Da qui nasce l'idea del Prof. Grassi di abbandonare la piattaforma Matlab e creare una toolbox simile accessibile da un qualunque browser web. Dopo aver contattato il Professor Mauro Migliardi, del Dipartimento di Ingegneria dell'Informazione dell'Università di Padova, è iniziata la ricerca di tesisti per l'implementazione della web app.
L'applicazione, in questo momento, mette a disposizione dell'utente sei tipi di test:

- Pure Tone Intensity Discrimination, che stima la soglia differenziale per le intensità sonore;
- Pure Tone Frequency Discrimination, che stima la soglia differenziale per le frequenze sonore;
- Pure Tone Duration Discrimination, che stima la soglia differenziale per le durate sonore;
- White Noise Amplitude Modulation Detection, che stima la sensibilità per la modulazione d'ampiezza;
- White Noise Gap Detection, che stima la capacità di rilevare un'interruzione in un suono;
- White Noise Duration Discrimination, che stima la soglia differenziale per le durate sonore, ma utilizzando un rumore bianco.

Per ciascuno di questi test, l'applicazione genera uno specifico numero n di stimoli (che possono essere suoni puri o rumori bianchi), di cui $n - 1$ stimoli sono uguali (li chiamiamo stimoli standard) e 1 ha una caratteristica differente (lo chiamiamo stimolo variabile) in base al tipo di test, e l'utente deve

essere in grado di riconoscere lo stimolo diverso.
Ad ogni iterazione, la differenza tra gli stimoli standard e lo stimolo variabile varia in base alle risposte precedenti dell'utente, secondo tre algoritmi:

- Simple Up Down
- Down One Up
- Three Down On Up

Per ogni esperimento possono essere impostate le caratteristiche dell'esperimento stesso, come il tipo di algoritmo, il numero di stimoli e il delta iniziale, e le caratteristiche degli stimoli, come la frequenza, la durata e l'intensità.
La web-app permette l'accesso sia ad utenti registrati che non. In particolare, entrambi possono effettuare esperimenti settati personalmente o accedere ad esperimenti creati da altri utenti tramite invite code. Gli utenti registrati hanno anche la possibilità di creare propri test da condividere con altri utenti.
Per quanto riguarda il lavoro effettivamente svolto durante la stesura della tesi, sono stati innanzitutto implementati gli ultimi tre test sopra citati. È stato, inoltre, cambiato il paradigma per la generazione degli stimoli e migliorata l'interfaccia grafica.

1

Toolbox

In questo capitolo viene spiegato in che cosa consiste il progetto della toolbox.

Il progetto riguarda la realizzazione di una web-app che permette all'utente di eseguire vari esperimenti e test per stimare la propria soglia psicofisica uditiva. L'utente può accedere a test settati da lui stesso o, tramite invite code, ad altri test creati da altri utenti. Questo permette a ricercatori e tesisti nell'ambito della psicoacustica di creare test da sottoporre ad altri soggetti, in modo da collezionare numerosi dati utili al loro studio.

Tutti gli esperimenti consistono fondamentalmente nel rilevare uno stimolo sonoro con una certa caratteristica (chiamato *variable stimulus*) presentato in mezzo ad un numero variabile di altri stimoli (chiamati *standard stimulus*). Attraverso l'analisi delle risposte, con vari algoritmi non parametrici (chiamati *adaptive staircase*), viene stimata la soglia uditiva psicofisica del soggetto, chiamata *threshold*, ovvero la minima differenza fisica tra suono standard e suono variabile che riesce ad essere rilevata.

Attualmente la toolbox offre sei tipi di test con tre diversi algoritmi per la stima della soglia.

1.1 Tipi di utenti

In questa sezione vengono elencati e spiegati i vari tipi di utenti della toolbox.

In base al tipo di utilizzo della toolbox, sono state ipotizzate tre tipologie di soggetti mappate in tre tipologie di utenti corrispondenti.

1.1.1 Guest

Un soggetto che desidera semplicemente provare la toolbox o che deve svolgere un esperimento creato da un altro utente, utilizzerà la toolbox come utente Guest. Un utente Guest è un utente che non ha effettuato il login. Questo tipo di utente è in grado di effettuare test personali o test creati da altri utenti registrati tramite invite code. Un esempio tipico di questo tipo di utente è una persona che viene invitata da un ricercatore di psicoacustica a svolgere un suo test tramite invite code.

1.1.2 Account Owner User

Un soggetto che desidera effettuare test personali o test da sottoporre ad altri utenti per poi consultarne i risultati, utilizzerà la toolbox come utente Account Owner User. Un utente Account Owner User è un utente che ha effettuato la registrazione ed è loggato nella toolbox. Questo tipo di utente ha a disposizione le stesse funzionalità di un utente Guest, ma ha anche la possibilità di creare nuovi test privati o da condividere ad altri utenti per poi visualizzarne e scaricarne i risultati. Un esempio tipico di questo tipo di utente è un ricercatore nell'ambito della psicoacustica che desidera raccogliere dati da analizzare.

1.1.3 Superuser

Un utente Superuser è un utente registrato che si occupa dell'amministrazione della toolbox. Ha pieno accesso a tutte le funzionalità e può visualizzare tutti gli account e i dati prodotti dalla toolbox. Un superuser può essere creato solamente da un altro superuser. Quindi, questo tipo di utente è rivolto a coloro che lavorano internamente alla toolbox e si occupano del suo sviluppo.

1.2 Presentazione web-app

Di seguito viene mostrato il funzionamento della toolbox, attraverso la presentazione delle varie pagine che la compongono.

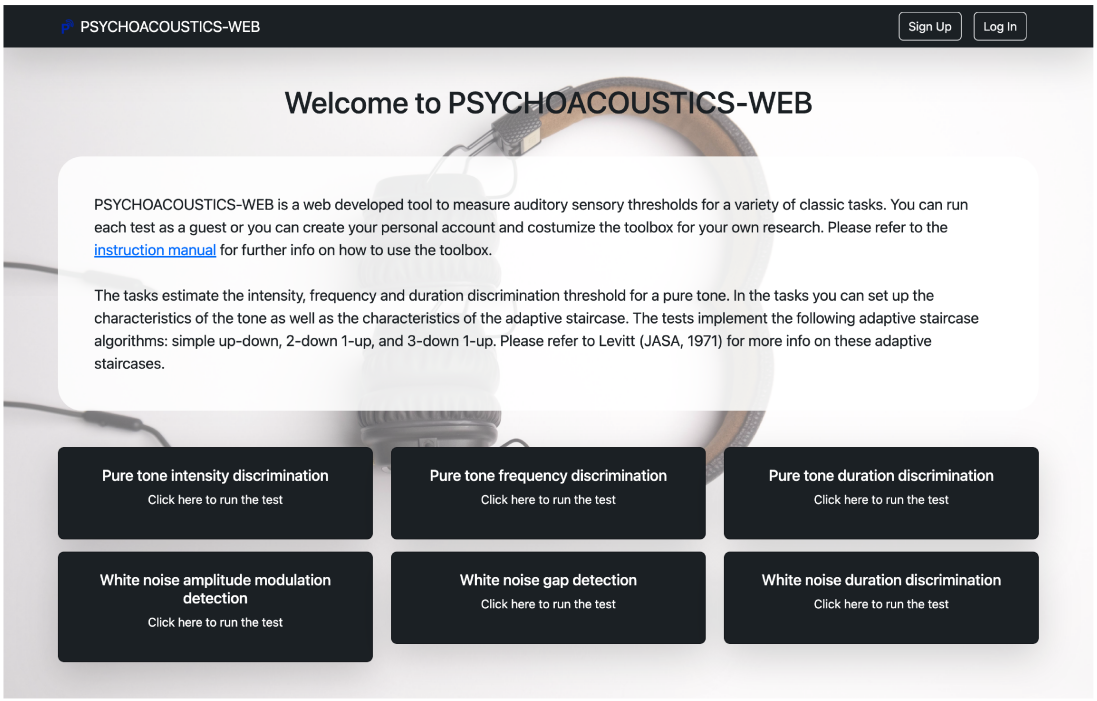

Figura 1.1: Homepage

Appena si accede al sito tramite il link http://psychoacoustics.dpg.psy.unipd.it ci si interfaccia con la homepage [figura 1.1]. Nella homepage è presente una breve descrizione in cui si leggono gli obiettivi della toolbox e come utilizzarla, e una griglia di bottoni per accedere ai vari tipi di test.
Nella homepage e in tutte le pagine principali è presente un menù che mostra bottoni diversi se l'utente è loggato o meno. Se l'utente non è loggato, è presente un bottone che porta alla pagina per la registrazione [figura 1.2] e un bottone che porta alla pagina per il login [figura 1.3]. Al contrario se l'utente è loggato, vengono mostrati bottoni per effettuare direttamente il log out, per accedere alla pagina in cui modificare le impostazioni dell'utente [figura 1.11] e per accedere alla pagina in cui visualizzare o scaricare i dati dei propri test [figura 1.12].

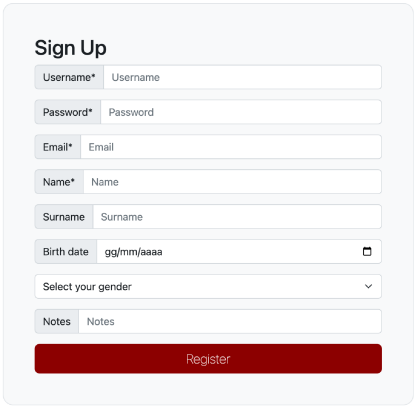

Figura 1.2: Pagina di registrazione

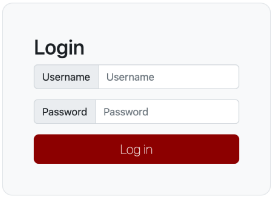

Figura 1.3: Pagina di login

Quando l'utente clicca su uno dei bottoni dei vari test, viene reindirizzato alla pagina in cui vengono chieste le informazioni base personali per accedere al test [figura 1.4]. Come il menù, questa pagina mostra informazioni diverse se l'utente è loggato o meno:

- se l'utente non è loggato, viene mostrato un form nel quale viene richiesto necessariamente il nome e non necessariamente il cognome, l'età, il genere ed eventuali note;
- se l'utente è loggato e vuole effettuare un test personale (quindi senza invite code), non sono richiesti ulteriori dati per proseguire;
- se l'utente è loggato e vuole effettuare un test tramite invite code, appena questo viene digitato, vengono richiesti gli stessi dati dell'utente non loggato.

La checkbox "Save results" indica se si vuole o meno salvare i risultati finali. Il bottone "Use mine" visibile solamente ad utenti loggati, permette di inserire il proprio invite code.

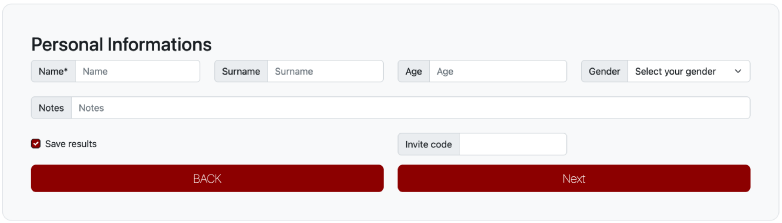

Figura 1.4: Pagina di informazioni utente

Se un utente accede ad un test tramite invite code, viene reindirazzato ad una pagina introduttiva [figura 1.5] in cui viene spiegato come svolgere il test e con una preview del test stesso.

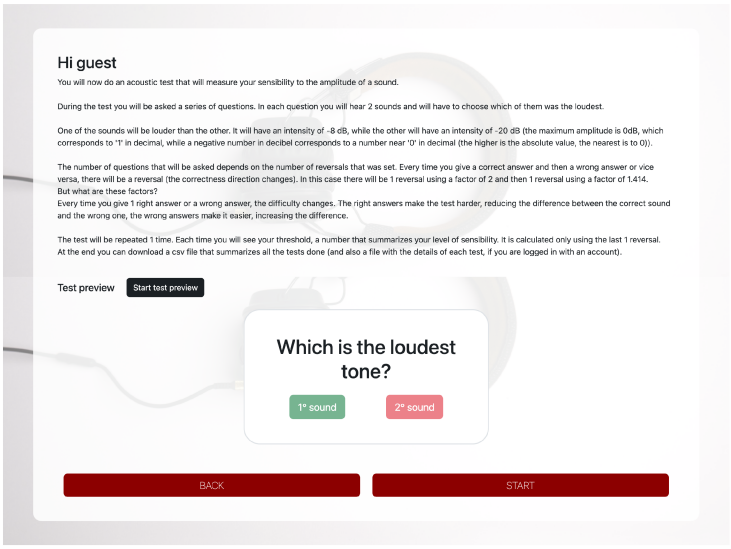

Figura 1.5: Pagina introduttiva test

Se un utente accede ad un test senza inserire invite code, viene reindirizzato ad una pagina in cui impostare le variabili in base al tipo di test. Per esempio, se il tipo di test coinvolge pure tone vengono richiesti frequenza, durata e ampiezza [figura 1.6], mentre se il tipo di test coinvolge white noise non viene richiesta la frequenza perché non avrebbe significato in questo contesto [figura 1.7]. In questa pagina, è presente un form composto da varie sezioni. Nella prima sezione vengono richieste le impostazioni del suono o rumore standard e sono differenti a seconda del tipo di test.
Se si tratta di test quali Pure Tone Intensity Discrimination [figura 1.6], Pure Tone Frequency Discrimination, Pure Tone Duration Discrimination, quindi di test riguardanti suoni puri, le impostazioni richieste sono:

- amplitude, indica l'intensità del suono standard espressa in dB;
- frequency, indica la frequenza del suono standard esperssa in Hz;
- duration, indica la durata del suono standard espressa in ms;
- duration onset ramp, indica la durata della rampa iniziale per evitare clipping;
- duration offset ramp, indica la durata della rampa finale per evitare clipping.

Se si tratta di test quali White Noise Duration Discrimination o White Noise Gap Detection, le impostazioni richieste sono le stesse dei test citati precedentemente con l'unica eccezione che la frequenza non è richiesta dato che non ha significato nel contesto di rumori bianchi. Se il test è White Noise Amplitude Modulation Detection [figura 1.7], le impostazioni richieste sono:

1. master settings:

- duration onset ramp;
- duration offset ramp;

2. carrier settings:

    - amplitude, indica l'intenisità del rumore da modulare espressa in Hz;
    - duration, indica la durata del rumore da modulare ms;

3. modulator settings, la funzione modulatrice è una funzione sinusoidale:

    - amplitude, indica l'ampiezza iniziale (in questo caso è il delta) della funzione modulatrice, cioè la profondità della modulazione, espressa in dB;
    - frequency, indica la frequenza della funzione modulatrice espressa in Hz;
    - phase, indica la fase iniziale della funzione modulatrice espressa in rad.

Nella seconda sezione vengono richieste informazioni riguardanti l'esperimento:

- n. of blocks, indica il numero di volte che il test deve essere ripetuto;
- nAFC, indica il numero di suoni o rumori da riprodurre;
- ITI, indica il tempo che intercorre prima della riproduzione del primo suono, espresso in ms;
- ISI, indica il tempo che intercorre tra un suono e l'altro, espresso in ms;
- delta, indica la differenza iniziale tra suoni standard e suono variabile, espresso in varie unità di misura a seconda del tipo di test.

Nella terza sezione vengono richieste informazioni riguardanti le caratteristiche dello staircase:

- radio button per decidere l'algoritmo da utlizzare:
    - SimpleUpDown;
    - TwoDownOneUp;
    - ThreeDownOneUp;

- checkbox "Feedback after response", indica se si vuole ricevere a schermo un feedback dopo la risposta se era giusta o sbagliata;

- checkbox ”Save settings”, visibile solamente se l’utente è loggato, indica se si vuole o meno salvare le impostazioni dell’intero test;
- first factor, indica il fattore per il delta da utilizzare per i primi reversal;
- first reversals, indica per quanti reversal viene utilizzato il primo fattore;
- second factor, indica il fattore per il delta da utilizzare per i secondi reversal;
- second reversals, indica per quanti reversal viene utilizzato il secondo fattore;
- reversal thresholdf, indica quanti reversal utilizzare per il calcolo della soglia.

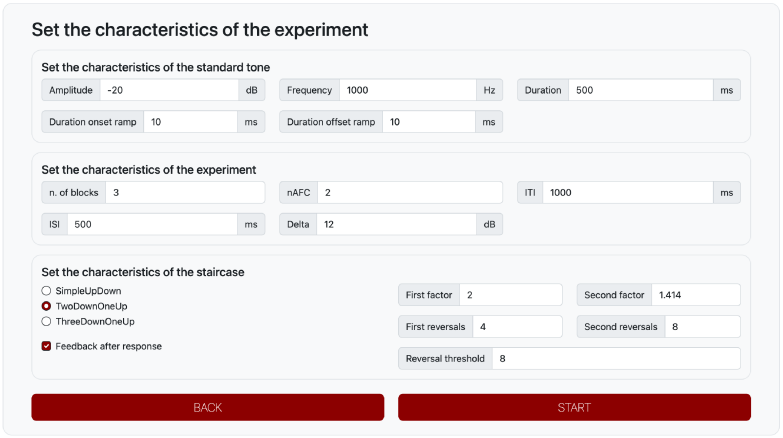

Figura 1.6: Pagina di impostazioni test ”Pure Tone Intensity Discrimination”

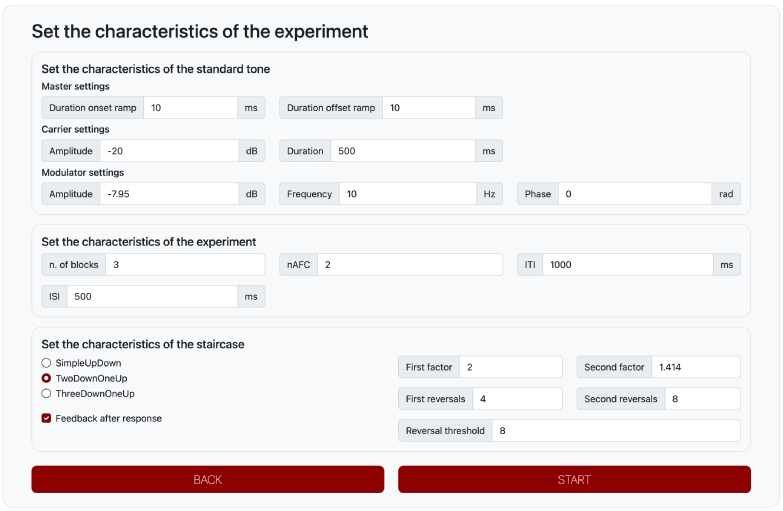

Figura 1.7: Pagina di impostazioni test ”White Noise Amplitude Modulation Detection”

Una volta effettuato l’accesso al test, l’utente viene reindirizzato alla pagina dove ha effettivamente inizio l’esperimento. Quando l’utente è pronto, può cliccare il pulsante ”Let’s start!” e iniziare il test

[figura 1.8]. Da questo momento in poi l'utente deve riconoscere il variable stymulus cliccando il pulsante corrispondente [figura 1.9].

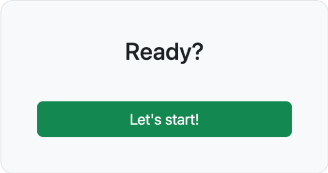

Figura 1.8: Pagina di inizio test

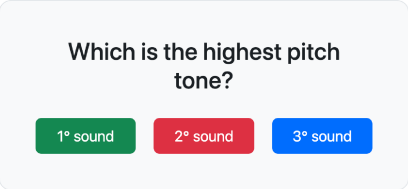

Figura 1.9: Pagina di test in corso

Alla fine dell'esperimento, l'utente viene reindirizzato ad una pagina in cui può scaricare in formato cvs i risultati di tutte le risposte cliccando il bottone "Download data" o solo le informazioni del test con il punteggio finale [figura 1.10].

Figura 1.10: Pagina di risultati finali

Cliccando sull'icona a forma di ingranaggio del menù, l'utente viene reindirizzato alla pagina mostrata in figura 1.11. In questa pagina sono presenti vari form.
Nel primo form, l'utente può copiare il proprio invite code o il link del sito tramite cui accedere al proprio test, in modo da condividerli velocemente. Può, inoltre, generare un nuovo invite code, disabilitando quello vecchio e può modificare le impostazioni del proprio esperimento.
Nel secondo form, visibile solamente se l'utente è superuser, si possono creare nuovi superuser, inserendo lo username. La password del nuovo superuser creato è inizialmente uguale al superuser che lo ha creato.
Nel terzo form, l'utente può modificare la propria password inserendo quella vecchia.
Nel quarto form, l'utente può modificare le informazioni relative al suo account, quali username, mail, nome, cognome, data di nascita, sesso ed eventuali note.

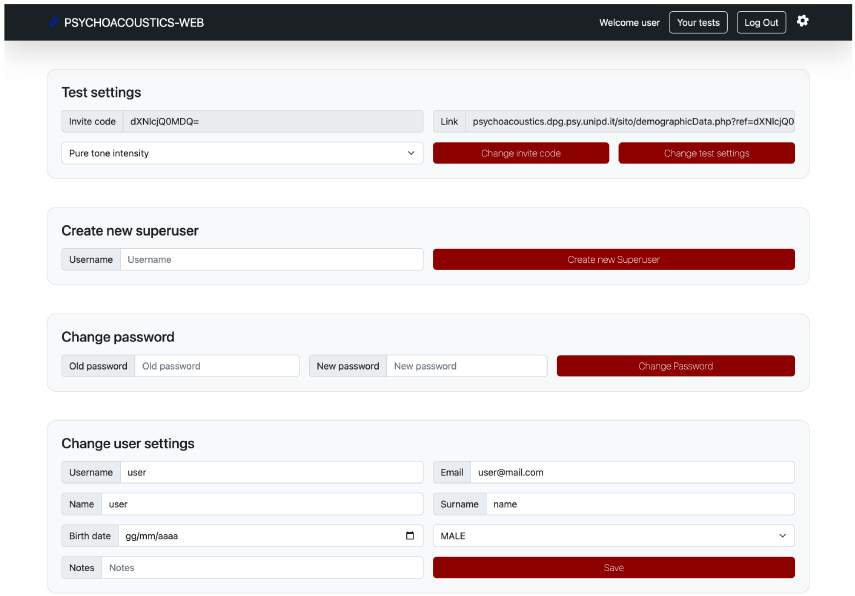

Figura 1.11: Pagina di impostazioni utente

Cliccando sul pulsante ”Your test” presente nel menù, l’utente viene reindirizzato alla pagina in cui visualizzare i test effettuati da lui o da altri utenti tramite il suo invite code. Da questa pagina, l’utente può scaricare in formato csv le informazioni riguardanti i test effettuati da lui o da altri utenti guest che hanno usato il suo invite code. Inoltre, se l’utente è superuser, può scaricare in formato csv tutto il contenuto del database.

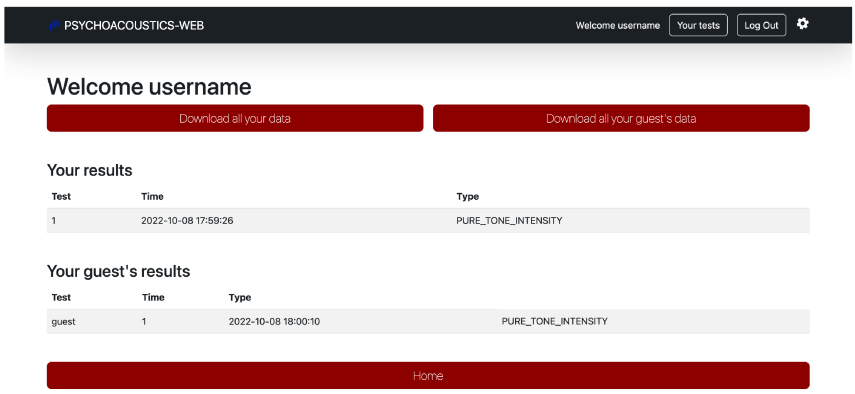

Figura 1.12: Pagina di visualizzazione test effettuati

1.3 Esperimenti presenti

In questa sezione vengono elencati e spiegati gli esperimenti presenti nella toolbox.

Un test consente ad un utente di stimare la propria soglia psicofisica uditiva, chiamata *threshold*.
La *detection threshold* è il minimo livello di stimolo rilevabile in assenza di altri stimoli dello stesso tipo, quindi, per esempio, White Noise Amplitude Modulation Detection consiste nel riconoscere il rumore modulato in ampiezza tra rumori non modulati in ampiezza.
La *discrimination threshold*, invece, è la minima differenza rilevabile tra due livelli di stimolo, quindi, per esempio, Pure Tone Duration Discrimination consiste nel riconoscere il suono con durata maggiore tra altri suoni con la stessa durata.
Un test è composto da una serie di iterazioni (chiamate *trial*). Ad ogni trial, lo stimolo variabile (chiamato *variable stimulus*) cambia il suo livello di stimolo, mentre gli altri stimoli (chiamati *standard stimulus*) rimangono invariati. Dopo ogni trial viene richiesto all'utente di riconoscere il variable stimulus.
Ad ogni trial, il variable stimulus varia in base alla correttezza delle risposte precedenti e all'algoritmo non parametrico utilizzato, chiamato *staircase*.
All'interno dell'analisi degli staircase, vengono chiamate *reversal* le inversioni della funzione monotona del livello di stimolo che, nel caso dei metodi adattivi implementati nella tool, si verificano quando dopo una serie di risposte corrette avviene una risposta sbagliata o viceversa.
Viene chiamata *delta* la misura della caratteristica differente tra variable e standard stimulus.

1.3.1 Pure Tone Intensity Discrimination

In questo tipo di test gli stimoli presentati sono dei suoni puri (chiamati *pure tone*), quindi dei segnali sinusoidali con una certa ampiezza, frequenza, fase (in questo caso non ci interessa) e durata. Ad ogni trial viene richiesto al soggetto di riconoscere il suono con intensità maggiore, cioè con ampiezza maggiore. Il delta, quindi, corrisponde alla differenza tra le ampiezze dello standard tone e del variable tone in decibel.

1.3.2 Pure Tone Frequency Discrimination

Per questo tipo di test gli stimoli utilizzati sono dei pure tone. Ad ogni trial il soggetto deve cercare di riconoscere il suono con tonalità più alta, cioè con frequenza maggiore. Quindi, per questo tipo di test, il delta corrisponde alla differenza tra le frequenze dello standard tone e del variable tone in hertz.

1.3.3 Pure Tone Duration Discrimination

Anche per questo tipo di test gli stimoli utilizzati sono dei pure tone. Ad ogni trial, al soggetto dell'esperimento è richiesto di riconoscere il suono con durata maggiore. Il delta per questo tipo di test è la differenza tra le durate dello standard tone e del variable tone in millisecondi.

1.3.4 White Noise Amplitude Modulation Detection

Per questo tipo di test gli stimoli presentati al soggetto sono dei rumori bianchi (chiamati *white noise*), cioè un rumore caratterizzato da ampiezza e durata. In questo caso, lo standard noise è un rumore bianco con una certa ampiezza e durata, mentre il variable noise è un rumore con la stessa ampiezza e durata ma modulato in ampiezza tramite una funzione modulatrice sinusoidale con una certa frequenza, ampiezza (chiamata anche profondità della modulazione) e fase. Quindi, ad ogni trial il soggetto deve cercare di rilevare il rumore modulato. Il delta per questo tipo di esperimenti è la profondità della modulazione in decibel.

1.3.5 White Noise Gap Detection

Per questo tipo di test gli stimoli utilizzati sono dei rumori bianchi. Lo standard noise è un rumore bianco con una certa ampiezza e durata e il variable noise è un rumore con la stessa ampiezza e durata ma con un breve momento di silenzio nel mezzo chiamato *gap*. Quindi, ad ogni trial l'utente deve riconoscere il rumore con il gap. Il delta in questo caso è la durata del gap in millisecondi.

1.3.6 White Noise Duration Discrimination

Questo tipo di test è del tutto simile al Pure Tone Duration Discrimination test, con l'unica differenza che non vengono presentati dei suoni puri ma dei campioni di rumore bianco.

1.4 Algoritmi per la stima della soglia

In questa sezione vengono elencati e spiegati gli algoritmi utilizzati nella toolbox per la stima della soglia psicofisica uditiva.

Durante il test, ad ogni trial il variable stimulus cambia il proprio livello di stimolo in base alla correttezza delle risposte precedenti e in base all'adaptive staircase utilizzato. In questo momento, la tool offre tre diversi algoritmi non parametrici per la stima della soglia, che sono SimpleUpDown, TwoDownOneUp e ThreeDownOneUp. Viene esposto inoltre l'algoritmo Method of Limits [sottosezione 1.4.1] per introdurre l'analisi degli staircase.

1.4.1 Method of Limits

Supponiamo di voler stimare la soglia psicofisica uditiva di discriminazione di una particolare grandezza fisica g (per esempio la frequenza). Al soggetto vengono quindi presentati due tipi di stimoli: il variable stimulus e lo standard stimulus. Il valore di g dello standard stimulus è fissato, mentre il valore di g del variable stimulus è più alto di un certo valore Δg. Il valore Δg varia durante i vari trial in funzione delle risposte del soggetto. Ad ogni trial, il soggetto deve quindi rilevare tra un determinato numero di stimoli, quello che ha il valore della grandezza fisica g più alto. Per ogni risposta corretta, il valore di Δg viene ridotto. In un certo trial n, il soggetto sbaglierà la risposta perché si suppone che si è oltrepassata la soglia psicofisica uditiva, causando un reversal. Quindi, la soglia viene calcolata facendo la media di Δg_n e Δg_{n-1}.

Questo metodo è stato ampiamente utilizzato per la sua semplicità e rapidità nel misurare la threshold, però non tiene conto dei falsi allarmi delle risposte. Inoltre, restituisce il livello di stimolo che corrisponde al 50% della funzione psicometrica, infatti restituisce la media tra il livello di stimolo dell'ultima risposta corretta (100% della funzione psicometrica) e il livello di stimolo della prima risposta sbagliata (0% della funzione psicometrica). Per questi motivi, la toolbox utilizza delle versioni aggiornate di Method of Limits.

1.4.2 SimpleUpDown

L'algoritmo SimpleUpDown risolve il problema dei falsi allarmi delle risposte del Method of Limits. In particolare, l'algoritmo SimpleUpDown non si ferma al primo reversal, ma continua a presentare al soggetto i due tipi di stimoli. Ad ogni risposta corretta il valore di Δg viene diviso per un certo fattore f, mentre ad ogni risposta sbagliata il valore di Δg viene moltiplicato sempre per f. Il fattore f

solitamente viene impostato ”grande” inizialmente per i primi reversal e viene impostato più ”piccolo” per i secondi. L'algoritmo non si ferma finché non avvengono un numero prestabilito di reversal. La threshold viene calcolata facendo la media degli ultimi n reversal, dove n è un numero prestabilito (nel nostro caso l'input ”reversal” visto precedentemente). Come il Method of Limits, anche l'algoritmo SimpleUpDown tiene conto del 50% della funzione psicometrica.

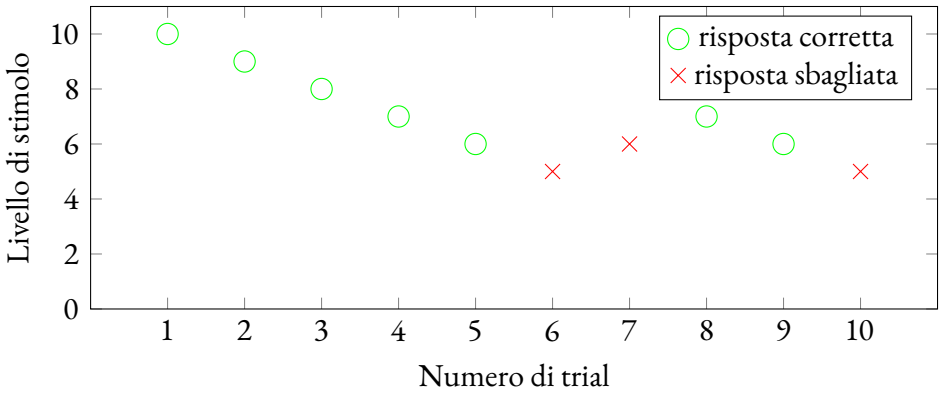

Figura 1.13: Threshold tracking SimpleUpDown

1.4.3 TwoDownOneUp

L'algoritmo TwoDownOneUp è molto simile all'algoritmo SimpleUpDown, infatti SimpleUpDown può essere chiamato OneDownOneUp. Quindi la differenza sostanziale tra questi algoritmi è che al posto di decrementare il valore di Δg per ogni risposta corretta, il valore di Δg viene decrementato solamente se si verificano due risposte corrette consecutive. Il valore di Δg viene incrementato per ogni risposta sbagliata. In questo modo, l'algoritmo TwoDownOneUp tiene traccia del 70.7% della funzione psicometrica.

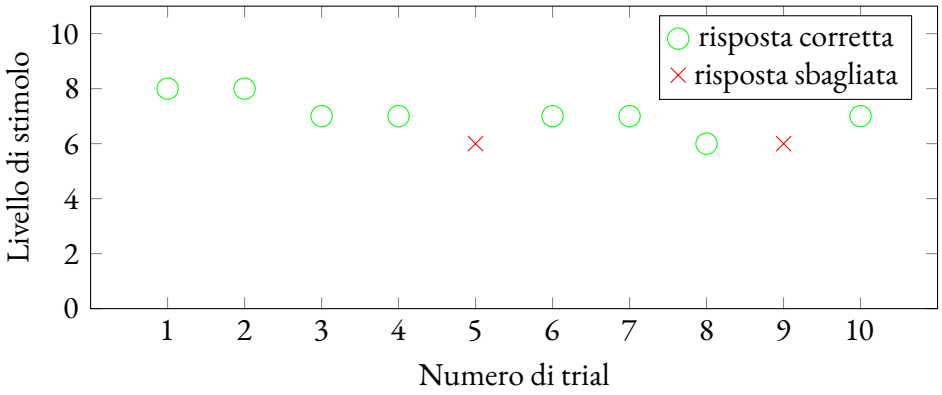

Figura 1.14: Threshold tracking TwoDownOneUp

1.4.4 ThreeDownOneUp

L'algoritmo ThreeDownOneUp è del tutto similare agli algoritmi precedentemente visti e come facilmente intuibile, l'unica differenza sta nel fatto che il valore di Δg viene decrementato solamente se si verificano tre risposte corrette consecutive. Come negli altri algoritmi, il valore di Δg viene incrementato per ogni risposta sbagliata. Quindi, l'algoritmo ThreeDownOneUp tiene traccia del 79.4% della funzione psicometrica.

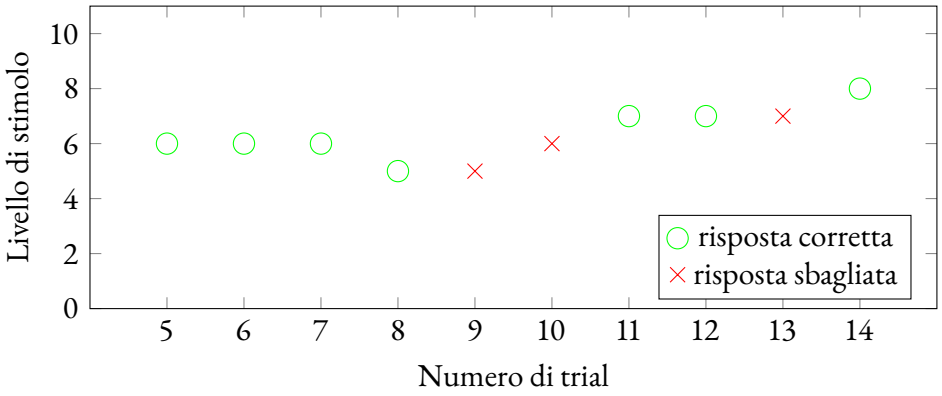

Figura 1.15: Threshold tracking ThreeDownOneUp

2

Progetto di tesi

In questo capitolo viene spiegato il lavoro effettivamente svolto per la stesura di questa tesi, analizzando il metodo di sviluppo e i linguaggi utilizzati.

Prima dell'inizio di questo progetto di tesi, la toolbox era già ad un buon punto di sviluppo. Era già stata implementata tutta la parte strutturale della toolbox, come login, creazione di esperimenti, accesso ad esperimenti tramite invite code, ecc. ecc. Erano già stati implementati tre tipi di test con i tre algoritmi citati nel capitolo precedente.

L'obiettivo del progetto di questa tesi è quindi quello di rendere la toobox pronta per il rilascio, aggiungendo nuovi tipi di esperimenti e migliorando la generazione dei suoni.

2.1 Metodologia di sviluppo

La metodologia di sviluppo utilizzata per questo progetto di tesi si può definire una metodologia agile, in quanto prevedeva una comunicazione costante tra il team di lavoro e il team manager che, in questo caso, coincide con il committente del progetto.
A differenza del modello di sviluppo a cascata o di modelli più tradizionali, la metodologia agile utilizzata consiste nell'effettuare una riunione, in presenza o tramite la piattaforma Zoom, con il Professor Grassi con cadenza settimanale. Durante queste riunioni, si espone il lavoro svolto durante la settimana per la revisione da parte del Professor Grassi. Alla fine della riunione, il Professor Grassi espone i nuovi task da eseguire per la settimana successiva.

2.2 Linguaggi e controllo di versione

La toolbox è stata sviluppata con vari linguaggi di programmazione:

- come linguaggio lato server è stato utilizzato PHP [2] senza l'ausilio di alcun framework, per il login o registrazione di utenti, per la creazione di nuovi esperimenti e per il dialogo con il database;
- come linguaggio lato client è stato utilizzato JavaScript [3] con l'ausilio della libreria Web Audio API [4] per la generazione di suoni e rumori;
- come linguaggio di markup è stato utilizzato HTML [5];
- come linguaggio per lo stile è stato utilizzato CSS [6] con l'ausilio del framework Bootstrap [7];
- come DBMS è stato utilizzato MySQL [8].

Per il controllo di versione è stata creata una repository privata su GitHub, in modo che ciascun sviluppatore potesse lavorare su un proprio branch indipendente.

2.3 Integrazione nel team

Prima dell'inizio di questo progetto di tesi, lo sviluppo della toolbox era già incominciato. Due tesisti del dipartimento di Ingegneria dell'Informazione, infatti, si erano già occupati della realizzazione della parte più strutturale della toolbox. In particolare, avevano:

- eseguito l'analisi dei requisiti;

- creato una prima versione del database;
- implementato la registrazione e il login di utenti;
- implementato tre tipi test (Pure Tone Intensity Discrimination, Pure Tone Frequency Discrimination, Pure Tone Duration Discrimination), sia come test personale sia come test tramite invite code;
- sviluppato le due pagine per la visualizzazione dei propri test e per la modifica delle impostazioni dell'utente.

All'inizio di questo progetto di tesi si sono svolte due riunioni con tutto il team di sviluppo per chiarire vari accorgimenti tecnici e programmare i task da svolgere con l'obiettivo di ultimare la toolbox nel suo complesso.

2.4 Implementazioni

In questa sezione vengono esposte le varie implementazioni effettuate durante questo progetto di tesi.

2.4.1 Cambio paradigma generazione stimoli

Come già ripetuto, durante i vari esperimenti l'utente deve rilevare lo stimolo variabile tra altri stimoli standard. Per generare e riprodurre i vari stimoli si è utilizzata la libreria JavaScript Web Audio Api.
La libreria Web Audio API gestisce le varie operazioni audio all'interno di un contesto audio e consente il routing modulare. Le varie operazioni audio vengono eseguite tramite dei nodi audio che collegati tra di loro formano un grafo di routing. Il design modulare di questa libreria consente la creazione di funzioni audio complesse con effetti dinamici. I nodi audio sono collegati in reti semplici tramite i loro input e output. Ogni nodo audio possiede una o più sorgenti. Una sorgente non è altro che un array di campioni che formano il segnale audio. Questi campioni possono essere calcolati matematicamente, importati da file o creati tramite funzioni di libreria.
Un tipico esempio di grafo di routing audio è formato da un nodo oscillatore (oscillatorNode) con una qualsiasi forma d'onda, collegato con un nodo di guadagno (gainNode) con un qualsiasi valore $a \in [0, 1]$, collegato con la destinazione del contesto (BaseAudioContext.destination) che indirizza l'audio alla scheda audio.
All'inizio di questo progetto di tesi, le sorgenti dei vari nodi audio venivano generate attraverso funzioni di libreria, ma questo approccio si è rilevato svantaggioso e quindi si è deciso di generare le sorgenti dei vari nodi audio matematicamente.
Un task importante all'inizio di questo progetto di tesi è stato rimuovere il clipping in entrata e in uscita dei vari pure tone. Il clipping si verificava perché il guadagno del segnale da riprodurre cambiava istantaneamente da 0 al valore del volume impostato. Per risolvere questo problema sono state inserite due rampe, *onset ramp* e *offset ramp*, per rallentare il cambio di valore del guadagno. Per implementare le due rampe è stata utilizzata inizialmente la funzione di libreria setTargetAtTime con parametri:

- target, il valore finale da raggiungere;
- startTime, il momento in cui deve iniziare la transizione;
- timeCostant, quanti secondi deve durare la transizione.

Questo approccio, però, ha una criticità importante: la funzione di transizione è una funzione esponenziale e ciò significa che il valore non raggiunge mai il target completamente.

Un altro task di questo progetto di tesi è stato aggiungere il tipo di test White Noise Amplitude Modulation Detection. Il problema principale per questo tipo di test è stato l'implementazione della modulazione d'ampiezza di un rumore.
Inizialmente, venivano collegati vari nodi audio per formare un grafo di routing audio:

- carrier, con sorgente un array con numeri generati casualmente tra -1 e $+1$;
- modulator, con sorgente una sinusoide generata tramite funzioni di libreria;
- master

Questo approccio però non dà la possibilità di avere una fase di modulazione.
Per questi motivi, si è deciso di cambiare paradigma per la generazione degli stimoli. Le sorgenti dei nodi non vengono più generate tramite funzioni di libreria, ma vengono generate matematicamente.

```
function playSound(soundAmplitude, soundFrequency, soundDuration, onsetRamp, offsetRamp) {
    var context = new AudioContext();   // nuovo contesto

    var volume = context.createGain();  // nuovo nodo di guadagno
    volume.gain.setValueAtTime(0, context.currentTime); // guadagno iniziale
    volume.gain.setTargetAtTime(soundAmplitude, context.currentTime, onsetRamp); // rampa iniziale
    volume.gain.setTargetAtTime(0, context.currentTime + soundDuration - 3 * offsetRamp, offsetRamp);   // rampa finale

    var oscillator = context.createOscillator();    // nuovo nodo oscillatore
    oscillator.frequency.value = soundFrequency;    // frequenza del suono
    oscillator.type = "sine";   // tipi di onda del suono
    oscillator.connect(volume); // connessione nodo oscillatore con nodo di guadagno
    volume.connect(context.destination);    // connessione nodo di guadagno con scheda audio
    oscillator.start(context.currentTime);  // inizio del suono
    oscillator.stop(context.currentTime + soundDuration);   // fine del suono
}
```

Code snippet 2.1: Paradigma iniziale per la generazione di un pure tone

```
function playSound(soundAmplitude, soundFrequency, soundDuration, onsetRamp, offsetRamp) {
    var context = new AudioContext();   // nuovo contesto

    var volume = context.createGain();  // nuovo nodo di guadagno
    volume.gain.value = 1;  // valore del guadagno a 1

    var channels = 1;   // numero canali di uscita
    var frameCount = context.sampleRate * soundDuration;    // imposto una durata massima del suono di soundDuration secondi
    var soundBuffer = context.createBuffer(channels, frameCount, context.sampleRate);   // creo un nuovo buffer
    let ramp = [];
    for (let channel = 0; channel < channels; channel++) {  // riempio il buffer con i campioni del suono
        let nowBuffering = soundBuffer.getChannelData(channel);
        for (let i = 0; i < frameCount; i++) {
            t = i / context.sampleRate;
            if (t < onsetRamp) {
                ramp[i] = (1 + Math.sin((t * Math.PI / onsetRamp) - (Math.PI / 2))) / 2;    // onset ramp
            } else if (t > soundDuration - offsetRamp) {
                ramp[i] = (1 + Math.sin(((t - (soundDuration - offsetRamp)) * Math.PI / offsetRamp) + (Math.PI / 2))) / 2;  // offset ramp
            } else {
                ramp[i] = 1;    // central zone
            }
            nowBuffering[i] = soundAmplitude * Math.sin(2 * Math.PI * soundFrequency * t) * ramp[i]; // t = i / context.sampleRate
        }
```

```
    }
    source = context.createBufferSource();  // creo nodo con sorgente buffer
    source.buffer = soundBuffer;    // imposto il buffer del nodo
    source.connect(volume); // connessione nodo con sorgente buffer con nodo di guadagno
    volume.connect(context.destination);    // connessione nodo di guadagno con scheda audio
    source.start(context.currentTime);  // inizio del suono
    source.stop(context.currentTime + soundDuration);   // fine del suono
}
```

Code snippet 2.2: Paradigma finale per la generazione di un pure tone

Come si può notare dal code snippet 2.2, definite r_i la rampa di ingresso, r_o la rampa di uscita e d la durata dello stimolo, la funzione matematica utilizzata per implementare le rampe è:

$$r(t) = \begin{cases} \frac{1}{2}\left(1 + \sin\left(\frac{\pi}{r_i}t - \frac{\pi}{2}\right)\right) & \text{se } t \in [0, r_i] \\ 1 & \text{se } t \in [r_i, d - r_o] \\ \frac{1}{2}\left(1 + \sin\left(\frac{\pi}{r_o}\left(t - (d - r_o)\right) + \frac{\pi}{2}\right)\right) & \text{se } t \in [d - r_o, d] \\ 0 & \text{altrimenti} \end{cases} \tag{2.1}$$

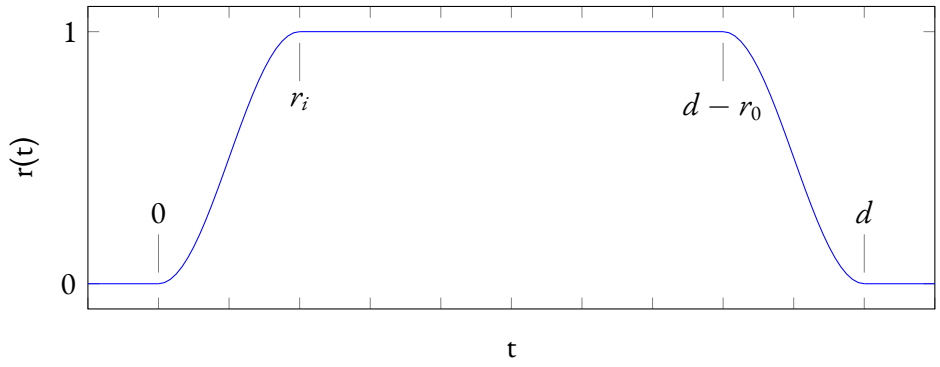

Figura 2.1: Funzione rampa

2.4.2 Test aggiunti

Durante questo progetto di tesi sono stati aggiunti alla toolbox tre ulteriori tipi di esperimenti con gli stessi tre algoritmi per la stima della soglia psicofisica uditiva.

White Noise Duration Discrimination

In questo tipo di test, il soggetto deve riconoscere il rumore bianco con durata maggiore. Quindi in questo caso il delta è la differenza tra le durate del rumore variabile e del rumore standard.
Per generare un rumore bianco è sufficiente riempire il buffer con dei valori casuali $v \in [0, 1]$. Quindi

il codice che la tool utilizza per generare rumori bianchi è il code snippet 2.3.

L'unico parametro della funzione playNoise che cambia per generare rumori standard e rumori variabili è noiseDuration; infatti per generare il rumore variabile viene passato come parametro noiseDuration la durata del rumore standard sommata al delta corrente.

```
function playNoise(noiseAmplitude, noiseDuration, onsetRamp, offsetRamp) {
    var context = new AudioContext();   // nuovo contesto
    var volume = context.createGain();  // nuovo nodo di guadagno
    volume.gain.value = 1;  // valore del guadagno a 1
    var channels = 1;   // numero canali di uscita
    var frameCount = context.sampleRate * noiseDuration;    // imposto una durata massima del suono di noiseDuration secondi
    var noiseBuffer = context.createBuffer(channels, frameCount, context.sampleRate);   // creo un nuovo buffer
    let ramp = [];
    for (let channel = 0; channel < channels; channel++) {  // riempio il buffer con i campioni del rumore
        let nowBuffering = noiseBuffer.getChannelData(channel);
        for (let i = 0; i < frameCount; i++) {
            t = i / context.sampleRate;
            if (t < onsetRamp) {
                ramp[i] = (1 + Math.sin((t * Math.PI / onsetRamp) - (Math.PI / 2))) / 2;    // onset ramp
            } else if (t > noiseDuration - offsetRamp) {
                ramp[i] = (1 + Math.sin(((t - (noiseDuration - offsetRamp)) * Math.PI / offsetRamp) + (Math.PI / 2))) / 2;  // offset ramp
            } else {
                ramp[i] = 1;    // central zone
            }
            nowBuffering[i] = noiseAmplitude * (Math.random() * 2 - 1) * ramp[i]; // riempio il buffer con valori casuali
        }
    }
    source = context.createBufferSource();  // creo nodo con sorgente buffer
    source.buffer = noiseBuffer;    // imposto il buffer del nodo
    source.connect(volume); // connessione nodo con sorgente buffer con nodo di guadagno
    volume.connect(context.destination);    // connessione nodo di guadagno con scheda audio
    source.start(context.currentTime);  // inizio del rumore
    source.stop(context.currentTime + soundDuration);   // fine del rumore
}
```

Code snippet 2.3: Generazione rumore bianco

White Noise Gap Detection

In questo tipo di test il soggetto deve riconoscere il rumore bianco in cui è presente un breve periodo di silenzio nel mezzo chiamato gap. Quindi il delta per questo tipo di test è il gap.

Per generare il rumore bianco standard si utilizza la stessa funzione playNoise del code snippet 2.3. Per generare il rumore variabile, invece, si utilizza una funzione differente playGapNoise, visibile nel code snippet 2.4.

Si può notare che per implementare il silenzio nel mezzo, i valori centrali dell'array ramp vengono impostati a 0. Nella transizione centrale (da 0 a 1 e viceversa) sono presenti due ulteriori rampe sinusoidali per evitare il clipping.

```
function playGapNoise(noiseAmplitude, noiseDuration, onsetRamp, offsetRamp, gap) {
    var context = new AudioContext();   // nuovo contesto
    var volume = context.createGain();  // nuovo nodo di guadagno
    volume.gain.value = 1;  // valore del guadagno a 1
    var channels = 1;   // numero canali di uscita
```

```
    var frameCount = context.sampleRate * noiseDuration;    // imposto una durata massima del suono di noiseDuration secondi
    var noiseBuffer = context.createBuffer(channels, frameCount, context.sampleRate);   // creo un nuovo buffer
    let ramp = [];
    for (let channel = 0; channel < channels; channel++) {  // riempio il buffer con i campioni del rumore
        let nowBuffering = noiseBuffer.getChannelData(channel);
        for (let i = 0; i < frameCount; i++) {
            t = i / context.sampleRate;
            if (t < onsetRamp) {
                ramp[i] = (1 + Math.sin((t * Math.PI / onsetRamp) - (Math.PI / 2))) / 2;    // onset ramp
            } else if (t >= onsetRamp && t < ((soundDuration / 2) - (gap / 2))) {
                ramp[i] = 1; // first central zone
            } else if (t >= ((soundDuration / 2) - (gap / 2)) && t < ((soundDuration / 2) - (gap / 2) + betweenRamp) && t < (soundDuration / 2)) {
                ramp[i] = (1 + Math.sin(((t - ((soundDuration / 2) - (gap / 2))) * Math.PI / betweenRamp) + (Math.PI / 2))) / 2; // offset ramp gap
            } else if (t >= ((soundDuration / 2) - (gap / 2) + betweenRamp) && t < ((soundDuration / 2) + (gap / 2) - betweenRamp)) {
                ramp[i] = 0; // central zone in gap
            } else if (t >= ((soundDuration / 2) + (gap / 2) - betweenRamp) && t < (soundDuration / 2) + (gap / 2)) {
                ramp[i] = (1 + Math.sin(((t - ((soundDuration / 2) + (gap / 2) - betweenRamp)) * Math.PI / betweenRamp) - (Math.PI / 2))) / 2; // onset
        ramp gap
            } else if (t >= (soundDuration / 2) + (gap / 2) && t < soundDuration - offsetRamp) {
                ramp[i] = 1; // second central zone
            } else if (t >= soundDuration - offsetRamp) {
                ramp[i] = (1 + Math.sin(((t - (soundDuration - offsetRamp)) * Math.PI / offsetRamp) + (Math.PI / 2))) / 2; // offset ramp
            } else {
                console.log("errore");
            }
            nowBuffering[i] = noiseAmplitude * (Math.random() * 2 - 1) * ramp[i]; // riempio il buffer con valori casuali
        }
    }
    source = context.createBufferSource();  // creo nodo con sorgente buffer
    source.buffer = noiseBuffer;    // imposto il buffer del nodo
    source.connect(volume); // connessione nodo con sorgente buffer con nodo di guadagno
    volume.connect(context.destination);    // connessione nodo di guadagno con scheda audio
    source.start(context.currentTime);  // inizio del rumore
    source.stop(context.currentTime + soundDuration);   // fine del rumore
}
```

Code snippet 2.4: Generazione rumore bianco con gap

Definite r_i la durata della rampa di ingresso, r_o la durata della rampa di uscita, d la durata del rumore, g la durata del gap e r_b la durata delle rampe nel gap, allora:

- i punti di inizio e fine della rampa di ingresso sono $r_{is} = 0$ e $r_{ie} = r_i$;
- i punti di inizio e fine della prima rampa del gap sono $r_{b1s} = \frac{d-g}{2}$ e $r_{b1e} = \frac{d-g}{2} + r_b$;
- i punti di inizio e fine della seconda rampa del gap sono $r_{b2s} = \frac{d+g}{2} - r_b$ e $r_{b2e} = \frac{d+g}{2}$;
- i punti di inizio e fine della rampa di uscita sono $r_{os} = d - r_o$ e $r_{oe} = d$;

Quindi la funzione matematica utilizzata per implementare il gap è:

$$r(t) = \begin{cases} \frac{1}{2}\left(1 + \sin\left(\frac{\pi}{r_i}(t - r_{is}) - \frac{\pi}{2}\right)\right) & \text{se } t \in [r_{is}, r_{ie}] \\ 1 & \text{se } t \in [r_{ie}, r_{b1s}] \\ \frac{1}{2}\left(1 + \sin\left(\frac{\pi}{r_b}(t - r_{b1s}) + \frac{\pi}{2}\right)\right) & \text{se } t \in [r_{b1s}, r_{b1e}] \\ 0 & \text{se } t \in [r_{b1e}, r_{b2s}] \\ \frac{1}{2}\left(1 + \sin\left(\frac{\pi}{r_b}(t - r_{b2s}) - \frac{\pi}{2}\right)\right) & \text{se } t \in [r_{b2s}, r_{b2e}] \\ 1 & \text{se } t \in [r_{b2e}, r_{os}] \\ \frac{1}{2}\left(1 + \sin\left(\frac{\pi}{r_o}(t - r_{os}) + \frac{\pi}{2}\right)\right) & \text{se } t \in [r_{os}, r_{oe}] \\ 0 & \text{altrimenti} \end{cases} \tag{2.2}$$

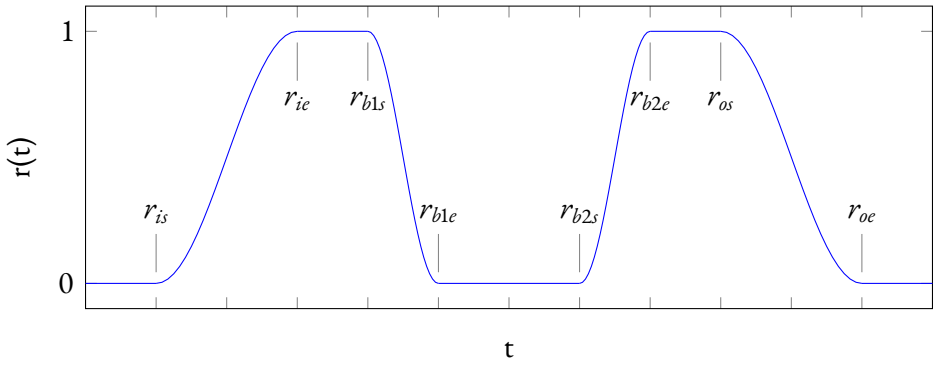

Figura 2.2: Funzione rampa per il gap

White Noise Amplitude Modulation Detection

In questo tipo di test il soggetto deve riconoscere il rumore bianco in cui è presente una modulazione d'ampiezza. Quindi il delta è la profondità della modulazione.
Per generare il rumore bianco standard viene chiamata la funzione playNoise del code snippet 2.3. Quindi, il problema principale da risolvere è stato creare la funzione per generare il rumore bianco variabile modulato in ampiezza.
Una prima versione di questa funzione utilizza il paradigma precedente per generare i suoni. In particolare, vengono creati un nodo carrier con buffer i campioni del rumore bianco e un nodo oscillatore con onda sinusoidale e frequenza pari alla frequenza di modulazione. Questi due nodi vengono collegati a due rispettivi nodi di guadagno e infine il nodo di guadagno del modulator viene collegato al nodo di guadagno del carrier. Questa versione, con questo metodo di generazione dei suoni, non dà

la possibilità di avere una fase di modulazione.

Quindi, si è deciso di cambiare paradigma e il codice utilizzato per generare il rumore variabile modulato è presente nel code snippet 2.5.

```
function playModulatedNoise(carrierAmplitude, carrierDuration, modulatorAmplitude, modulatorFrequency, modulatorPhase, onsetRamp, offsetRamp){
    var context = new AudioContext();   // nuovo contesto
    var volume = context.createGain();  // nuovo nodo di guadagno
    volume.gain.value = 1;   // valore del guadagno a 1
    let channels = 1;   // numero canali di uscita
    var frameCount = context.sampleRate * carrierDuration;  // imposto una durata massima del suono di noiseDuration secondi
    let noiseBuffer = context.createBuffer(channels, frameCount, context.sampleRate);   // creo un nuovo buffer
    for (let channel = 0; channel < channels; channel++) {  // riempio il buffer con i campioni del rumore
        let master = noiseBuffer.getChannelData(channel);
        let carrier = [];
        let modulator = [];
        let ramp = [];
        for (let i = 0; i < frameCount; i++) {
            t = i / context.sampleRate;
            if (t < onsetRamp) {
                ramp[i] = (1 + Math.sin((t * Math.PI / onsetRamp) - (Math.PI / 2))) / 2; // onset ramp
            } else if (t > carrierDuration - offsetRamp) {
                ramp[i] = (1 + Math.sin(((t - (carrierDuration - offsetRamp)) * Math.PI / offsetRamp) + (Math.PI / 2))) / 2; // offset ramp
            } else {
                ramp[i] = 1; // central zone
            }
            carrier[i] = carrierAmplitude * (Math.random() * 2 - 1);    // carrier
            modulator[i] = modulatorAmplitude * Math.sin(i * 2 * Math.PI * modulatorFrequency / context.sampleRate + modulatorPhase);   // modulator
            master[i] = (((carrierAmplitude / (carrierAmplitude + modulatorAmplitude)) + (modulator[i] / (carrierAmplitude + modulatorAmplitude))) *
      carrier[i]) * ramp[i];
        }
    }
    source = context.createBufferSource();  // creo nodo con sorgente buffer
    source.buffer = noiseBuffer;    // imposto il buffer del nodo
    source.connect(volume);    // connessione nodo con sorgente buffer con nodo di guadagno
    volume.connect(context.destination);   // connessione nodo di guadagno con scheda audio
    source.start(context.currentTime);   // inizio del rumore
    source.stop(context.currentTime + carrierDuration);   // fine del rumore
}
```

Code snippet 2.5: Generazione rumore bianco con modulazione

Vengono definiti:

- a_c, d_c rispettivamente l'ampiezza e la durata del segnale carrier $c(t)$;
- a_m, f_m, p_m rispettivamente l'ampiezza, la frequenza e la fase del segnale modulator $m(t)$

Quindi il segnale master $y(t)$ risultato della modulazione del segnale carrier $c(t)$ tramite il segnale modulator $m(t)$ è:

$$y(t) = \left(\frac{a_c}{a_c + a_m} + \frac{m(t)}{a_c + a_m} \right) c(t) \tag{2.3}$$

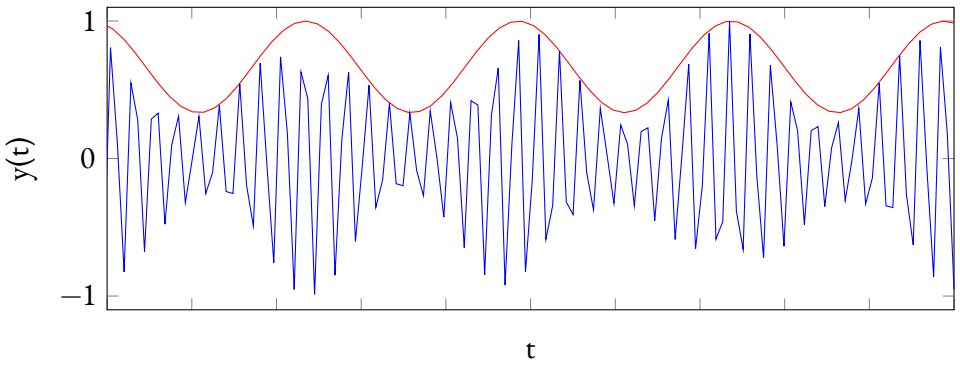

Figura 2.3: Funzione modulazione in ampiezza

2.4.3 Altre implementazioni

Durante questo progetto di tesi, sono stati svolti altri task più marginali per migliorare le funzionalità della toolbox e renderla disponibile al pubblico. Alcuni esempi disponibili in seguito.

Test preview

Uno dei primi task da svolgere è stato la realizzazione di una breve preview per i test eseguiti tramite invite code.
Quando un utente vuole eseguire un test tramite invite code, viene indirizzato in una pagina in cui viene esposta una breve descrizione dell'esperimento che andrà a svolgere con un'anteprima di prova. Questa anteprima è utile perché l'utente può prendere confidenza con le modalità di risposta e può verificare che l'audio si senta correttamente.
La pagina in cui è presente la preview dell'esperimento è visionabile nella figura 1.5.

Interfaccia grafica

È stata inoltre migliorata l'interfaccia grafica. Molte pagine della toolbox non erano correttamente responsive a causa di uno sbagliato utilizzo delle griglie di Bootstrap. Inoltre, venivano utilizzati stili diversi per vari oggetti che avevano la stessa funzione. Ad esempio, quasi tutti i form presenti avevano stili differenti.
Quindi, è stato migliorato il comportamento responsive di tutte le pagine in modo che siano correttamente navigabili attraverso qualsiasi dispositivo, sono stati aggiustati i vari allineamenti dei vari oggetti delle pagine e si è uniformato lo stile di tutte le pagine.
Ecco alcuni esempi di pagine con la versione precedente dell'interfaccia grafica.

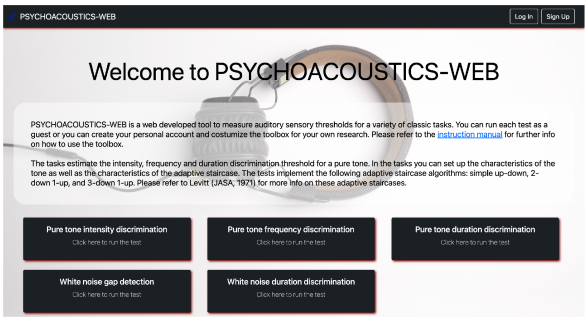

Figura 2.4: Versione precedente della homepage

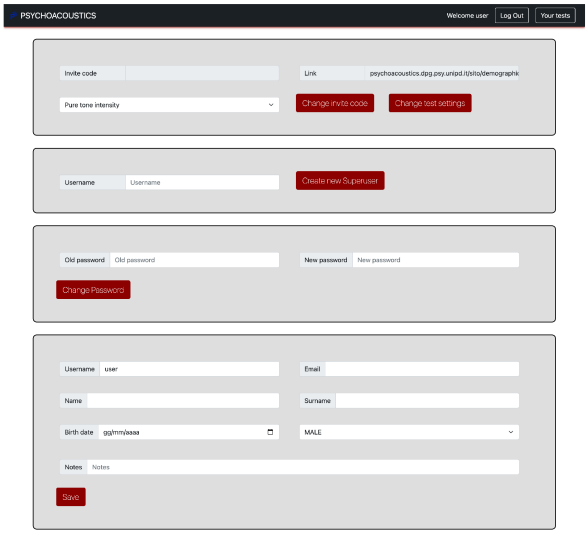

Figura 2.5: Versione precedente della pagina di impostazioni utente

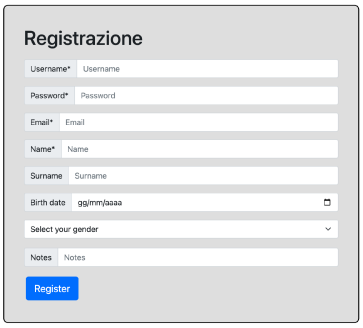

Figura 2.6: Versione precedente della pagina di registrazione

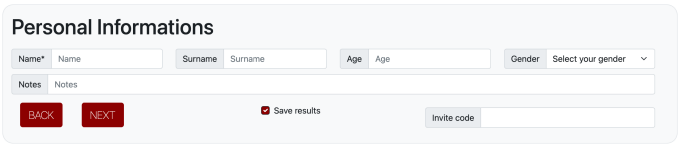

Figura 2.7: Versione precedente della pagina di informazioni utente

Contenuti testuali

Un altro task semplice ma comunque importante è stato lo spostamento dei contenuti testuali della toolbox, quali la descrizione presente in homepage o le descrizioni dei vari esperimenti, in un unico file esterno. In questo modo, sarà sempre possibile modificare i testi senza intervenire nei file php.
È stato utilizzato un file JSON [9] con le varie label uguali agli ID degli elementi HTML delle varie pagine. Quindi il codice per importare i testi dal file JSON è il code snippet 2.6.

```
fetch("files/texts.json").then(response => response.json()).then(data => {
    for (var attr in data) {
        element = document.getElementById(attr);
        if (element != null)
            element.innerHTML = data[attr];
    }
})
```

Code snippet 2.6: Import dei testi da file esterno

2.4.4 Debug

Durante questo progetto di tesi, sono stati corretti vari bug presenti nella toolbox. Uno fra questi è per esempio che l'utente registrato non riusciva a vedere i test effettuati da altri utenti tramite il suo invite code.
Verso la fine di questo progetto, insieme ad una studentessa del dipartimento di Psicologia Generale, è stato iniziato il debug completo dell'intera toolbox.
Il debug si sviluppa su due fronti: bisogna verificare che tutte le funzionalità della toolbox lavorino correttamente e bisogna verificare la corretta generazione dei vari stimoli.
Per verificare che tutte le funzionalità lavorino correttamente è necessario testare la toolbox come se fossimo degli utenti finali.
Per verificare la corretta generazione degli stimoli si sono adottati due metodi.
Solo per questa fase di testing, è stato aggiunto un pulsante nella pagina del test in corso che permette, ad ogni trial, di scaricare in formato csv tutti i parametri degli stimoli con i relativi samples.
Altrimenti, attraverso alcune estensioni per il browser (in questo caso si è utilizzato Chrome Audio Capture su browser Google Chrome) è possibile registrare l'audio riprodotto da uno specifico tab del browser e salvarlo in formato .mp3 o .wav.
Attraverso questi due metodi, i dati possono essere verificati tramite un qualsiasi audio editor come ad esempio Audacity.

2.5 Database

È stata infine modificata leggermente la struttura del database in modo da poter contenere le nuove informazioni aggiunte come le durate delle rampe o le informazioni per il test White Noise Amplitude Modulation Detection.

Lo schema logico relazionale della corrente versione del database della toolbox è il seguente:

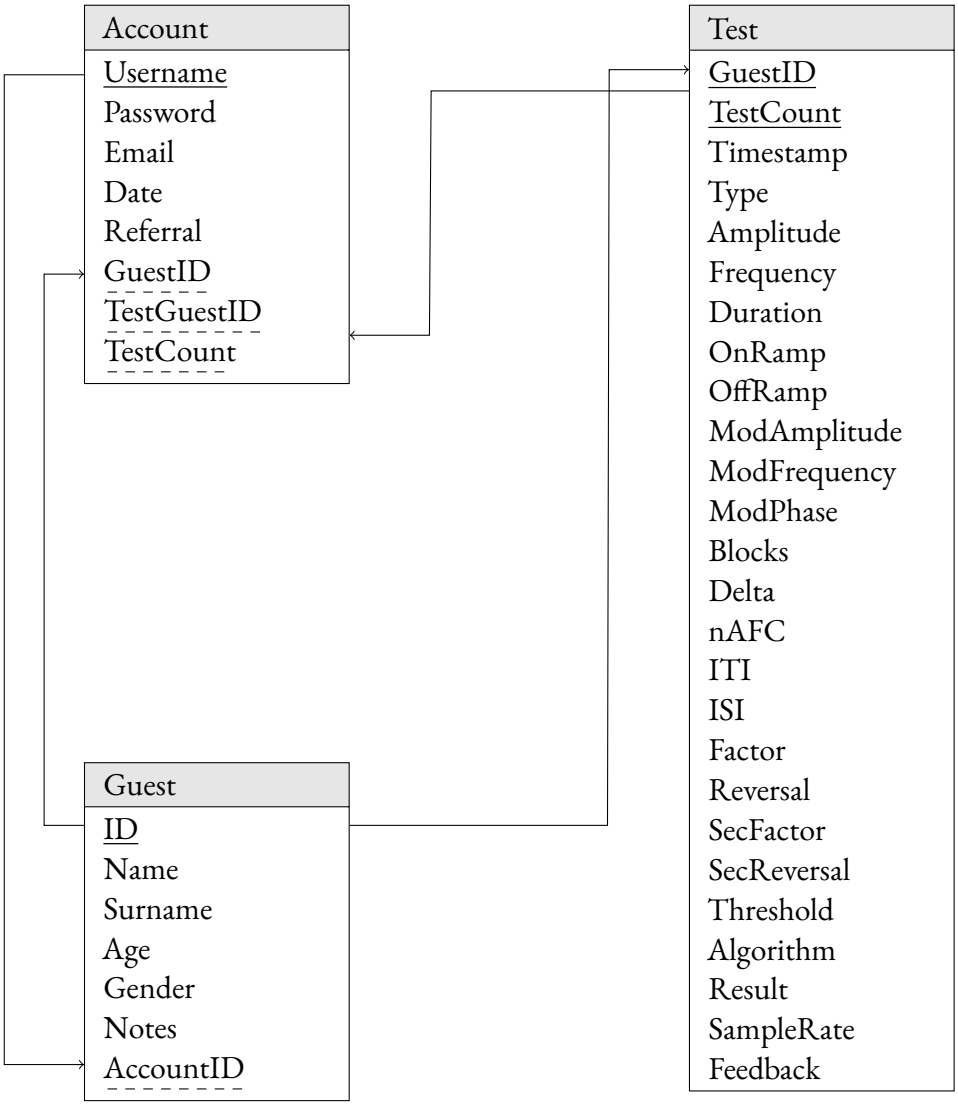

Figura 2.8: Schema logico relazione database

Si può notare facilmente che il database è strutturato in tre tabelle collegate attraverso quattro associazioni. I nomi delle tabelle e degli attributi sono abbastanza autoesplicativi. [10]

Le tabelle sono:

- la tabella Account contiene tutti i dati relativi agli utenti che si registrano, un account è identificato da uno username;

- la tabella Guest contiene i dati relativi agli utenti registrati o meno che effettuano un test, un guest è identificato tramite un ID auto-incrementale per gestire casi di omonimia;
- la tabella test contiene i dati di tutti i test effettuati, un test è identificato dalla combinazione dell'ID del guest che lo ha svolto e dal numero progressivo di test che il guest ha svolto.

Le associazioni sono:

- l'associazione dalla tabella Guest alla tabella Test, in quanto un test è identificato esternamente attraverso l'ID del guest che lo ha svolto;
- l'associazione dalla tabella Test alla tabella Account, utile per memorizzare le impostazioni di default del test per un account appena creato;
- l'associazione dalla tabella Guest alla tabella Account, utile per collegare l'account con il relativo ID del guest;
- l'associazione dalla tabella Account alla tabella Guest, utile per verificare se un guest utilizza l'invite code di un account.

Conclusioni

Durante la lettura di questa tesi, si può facilmente intuire che la toolbox è uno strumento innovativo e davvero utile nell'ambito degli esperimenti di psicoacustica che potrebbe effettivamente essere utilizzata efficacemente da numerosi ricercatori.

La conclusione di questo progetto non preclude di certo la possibilità di aggiungere ulteriori funzionalità. Ad esempio, come discusso con il Professor Grassi, si potrebbe dare la possibilità agli utenti di programmare nuove tipologie di test per poi renderle disponibili a tutti.

Arrivati alla fine di questo percorso, si può affermare che l'obiettivo prefissato di rendere la toolbox pronta per il pubblico sia stato raggiunto. Ovviamente, solo con il rilascio completo e quindi l'effettivo utilizzo da parte del pubblico si potranno scoprire eventuali errori da risolvere o possibili migliorie da implementare.

Bibliografia

[1] A. S. Massimo Grassi, "Mlp: A matlab toolbox for rapid and reliable auditory threshold estimation," *Behavior Research Methods*, 2009.

[2] Php. [Accessed: 4/11/2022]. [Online]. Available: https://www.php.net/

[3] Javascript. [Accessed: 4/11/2022]. [Online]. Available: https://www.javascript.com/

[4] Webaudioapi. [Accessed: 4/11/2022]. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

[5] Html. [Accessed: 4/11/2022]. [Online]. Available: https://www.w3schools.com/html/

[6] Css. [Accessed: 4/11/2022]. [Online]. Available: https://www.w3schools.com/css/

[7] Bootstrap. [Accessed: 4/11/2022]. [Online]. Available: https://getbootstrap.com/

[8] Mysql. [Accessed: 4/11/2022]. [Online]. Available: https://www.mysql.com/

[9] Json. [Accessed: 4/11/2022]. [Online]. Available: https://www.json.org/

[10] P. Atzeni, *Basi di dati*, 5th ed. Milano: McGraw-Hill Education srl, 2018.

Ringraziamenti

Arrivati alla fine di questo lavoro, ci terrei a ringraziare alcune delle persone presenti in questo percorso.

Innanzitutto, mi preme ringraziare il mio relatore Professor Mauro Migliardi e il mio correlatore Professor Massimo Grassi per la loro disponibilità, ma anche e soprattutto per avermi dato l'opportunità di vedere i miei studi applicati a un ambito differente.

Ringrazio la mia famiglia, in particolare Manuel, Chiara, Andrea e Beatrice per il loro supporto e i miei genitori Sabrina e Antonello per avermi permesso di passare serenamente questi anni di università.

Ringrazio i miei amici, in particolare Andrei per i suoi preziosi consigli.

Infine, ringrazio Caterina per essermi sempre stata vicino e per aver sempre creduto in me.