0

TuneJS - Documentazione

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 5

2.3 Use case 5

2.4 Pianificazione 5

2.5 Analisi dei mezzi 5

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 6

3.2 Design dei dati e database 6

3.3 Design delle interfacce 6

3.4 Design procedurale 6

4 Implementazione 7

5 Test 7

5.1 Protocollo di test 7

5.2 Risultati test 8

5.3 Mancanze/limitazioni conosciute 8

6 Consuntivo 8

7 Conclusioni 8

7.1 Sviluppi futuri 8

7.2 Considerazioni personali 8

8 Glossario 8

9 Bibliografia 9

9.1 Bibliografia per articoli di riviste: 9

9.2 Bibliografia per libri 9

9.3 Sitografia 9

10 Allegati 9

# Introduzione

## Informazioni sul progetto

TuneJS di Michele Beccarini

Docente responsabile: Guido Montalbetti

I3BB, Sezione Informatica, Modulo 306

Inizio progetto: 05.09.25

Fine progetto: 01.01.25

## Abstract

*The current state of online tuners is nothing short of dreadful. They lack the most essential features, are cluttered with ads and useless text, and their interfaces are stuck 15 years in the past. With this project, TuneJS aims to provide a clean and modern interface with simple yet essential functions, so intuitive that even a 7-year-old could tune an instrument if their parents handed it to them.*

*Using pitch detection algorithms, TuneJS can automatically recognize the played string and will clearly indicate—both visually and in writing—whether the string should be tightened or loosened, helping users achieve the most accurate tuning possible.*

## *With TuneJS, tuning becomes a pleasure, not just an obligation before playing!*

## Scopo

Lo scopo di TuneJS è di permettere all’utente di accordare un qualsiasi strumento (modalità custom) in modo semplice e veloce, senza il bisogno di usare direttamente il computer: se l’utente suona la corda, il computer individua automaticamente la corda suonata e dice se bisogna allentare o tirarla. Una volta accordata, il computer rimarrà in attesa che l’utente suoni un’altra corda da accordare.

# Analisi

## Analisi del dominio

Il lavoro sarà sviluppato utilizzando HTML, CSS e Javascript. Devo prima capire come catturare l’audio del microfono in streaming, poi analizzarlo e utilizzando degli algoritmi di pitch detection. Ogni nota ha una frequenza misurata in Hz, che varia leggermente a dipendenza della nota di riferimento La. Lo standard globale sarebbe un La di 440Hz ma, come ogni accordatore che si rispetti, voglio che sia possibile impostare un valore custom. Alternative molto comuni sono 432Hz o simili. Questa cifra rappresenta quindi il valore di riferimento dell’accordatura. Per la rilevazione della corda suonata, utilizzerò un range di più o meno di 10/15hz. Per un riferimento queste sono i valori delle note di un accordamento standard di una chitarra a 6 corde in Hz con il La a 440:

E2 (Mi basso) ≈ 82.41 Hz

A2 (La) ≈ 110.00 Hz

D3 (Re) ≈ 146.83 H

G3 (Sol) ≈ 196.00 Hz

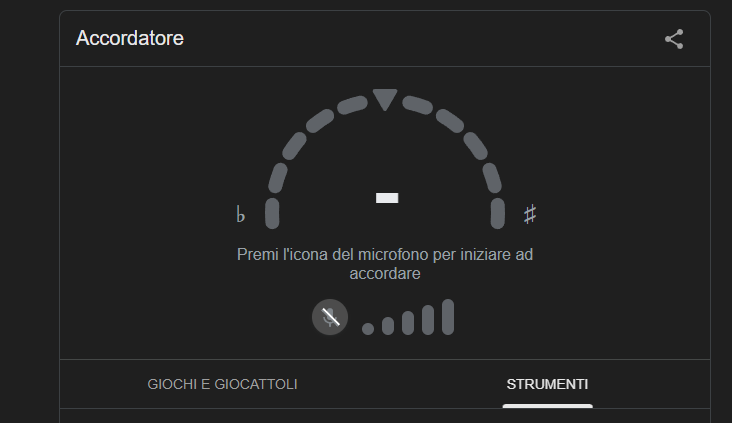
B3 (Si) ≈ 246.94 Hz

E4 (Mi acuto) ≈ 329.63 Hz

Per esempio, il range di individuazione del mi basso sarà circa tra 75Hz e 90Hz.   
Se invece voglio cambiare il La di riferimento, basta fare la proporzione rispetto alla nota in 440Hz. Per esempio, 82.41Hz in 440 sono

82.41x(432/440)≈80.88Hz

Si calcola facendo dei semplici rapporti.

Parlando dei prodotti simili, su internet ce ne sono di molti tipi diversi. Con una semplice ricerca su internet si può vedere che ce n’è uno pure ufficiale google integrato nel motore di ricerca.  
  


Oppure anche dei siti a parte dedicati come:

<https://www.musicca.com/it/accordatore>

<https://tuner-online.com/it/>

<https://theonlinemetronome.com/instrument-tuner>

per citarne qualcuno. Il problema è che sono tutti dei siti incasinati con informazioni inutili, pubblicità e mancanza di opzioni utili come la regolazione del La. Quindi il mio scopo sarebbe di fare un’applicazione web pulita semplice, senza nessuna informazione di troppo e con la giusta quantità di funzioni, descritte prima.

Si tratta quindi di un prodotto destinato a qualsiasi persona con uno strumento da accordare, che sia un professionista, un amatoriale, oppure qualcuno che ha semplicemente trovato una vecchia chitarra in soffitta e ha voglia di strimpellarla. La tipologia di utenti varia molto.

E la parte bella sta nel fatto che, sebbene ci voglia una certa conoscenza nell’ambito per realizzare l’applicativo, per utilizzarlo bisogna solo saper avvitare gli ingranaggi degli strumenti seguendo le indicazioni visive sullo schermo.

## Analisi e specifica dei requisiti

### Requisiti funzionali

* Rilevazione audio: Il sistema deve catturare in tempo reale il suono prodotto dal microfono dell’utente.
* Riconoscimento della nota: Deve identificare automaticamente quale corda è stata suonata, entro un margine di ±10/15 Hz.
* Indicazioni per l’accordatura: Il software deve indicare visivamente e testualmente se la corda deve essere tirata o allentata.
* Impostazione frequenza di riferimento: Deve essere possibile impostare il La di riferimento (default 440 Hz, opzioni custom come 432 Hz).
* Supporto strumenti: L’app deve consentire l’accordatura di almeno una chitarra a 6 corde e un ukulele a 4 corde, con possibilità di modalità custom per altri strumenti.
* Feedback continuo: Dopo aver accordato una corda, il sistema deve rimanere in ascolto per identificare la corda successiva senza necessità di ulteriori azioni da parte dell’utente.

### Requisiti non funzionali

* Interfaccia utente: Pulita, chiara e semplice, senza pubblicità o informazioni inutili.
* Compatibilità: Deve funzionare su browser moderni (Chrome, Firefox, Edge, Safari).
* Prestazioni: La latenza tra il suono e il feedback dell’applicativo deve essere inferiore a 200 ms.
* Sicurezza: L’accesso al microfono deve essere gestito in maniera sicura e trasparente, seguendo le linee guida del browser. Sarà tutto gestito localmente tramite javascript e non sarà inviato niente a nessun server
* Accessibilità: Il software deve poter essere utilizzato da utenti di qualsiasi età, anche senza conoscenze tecniche avanzate.

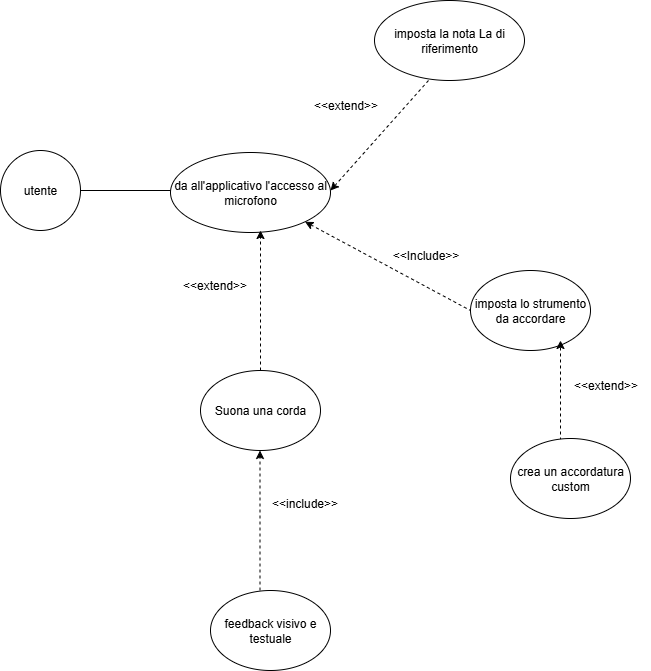
### Requisiti opzionali

* Memorizzazione accordature: Possibilità di salvare configurazioni personalizzate.
* Indicatori visivi avanzati: Aggiunta di grafici o colori per visualizzare quanto la corda è vicina alla frequenza corretta.
* Community di accordature: gli utenti possono pubblicare le proprie accordature dei propri strumenti oppure usare quelle degli altri già fatte.

### Spiegazione elementi tabella dei requisiti:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Nome | Priorità | Versione | Note | Sotto requisiti |
| RF01 | Rilevazione audio | Alta | 1.0 | Deve catturare il suono in tempo reale dal microfono dell’utente | Nessuno |
| RF02 | Riconoscimento della nota | Alta | 1.0 | Precisione ±10/15 Hz | Nessuno |
| RF03 | Indicazioni per accordatura | Alta | 1.0 | Feedback visivo e testuale su tirare o allentare la corda | Nessuno |
| RF04 | Impostazione frequenza di riferimento | Media | 1.0 | Default 440 Hz, opzioni personalizzate come 432 Hz | Nessuno |
| RF05 | Supporto strumenti | Alta | 1.0 | Deve funzionare per chitarra 6 corde e ukulele 4 corde | Chitarra 6 corde, Ukulele 4 corde, Modalità custom |
| RF06 | Feedback continuo | Media | 1.0 | Dopo accordatura corda, sistema rimane in ascolto | Nessuno |
| RNF01 | Interfaccia utente | Alta | 1.0 | Pulita, chiara, senza pubblicità | Nessuno |
| RNF02 | Compatibilità | Alta | 1.0 | Browser moderni (Chrome, Firefox, Edge, Safari) | Nessuno |
| RNF03 | Prestazioni | Media | 1.0 | Latenza <200 ms | Nessuno |
| RNF04 | Sicurezza | Alta | 1.0 | Gestione microfono locale, dati non inviati a server | Nessuno |
| RNF05 | Accessibilità | Media | 1.0 | Utilizzabile da utenti di qualsiasi età | Nessuno |
| RO01 | Memorizzazione accordature | Bassa | 1.0 | Possibilità di salvare configurazioni personalizzate | Nessuno |
| RO02 | Indicatori visivi avanzati | Bassa | 1.0 | Grafici o colori per visualizzare vicinanza alla frequenza corretta | Nessuno |
| RO03 | Community di accordature | Bassa | 1.0 | Pubblicazione e condivisione accordature | Nessuno |

## Use case



## Pianificazione

|  |
| --- |
|  |

## Analisi dei mezzi

### Software

* Chrome, Safari, Firefox, per la visualizzazione e il testing del html, css e javascript
* Microsoft Project, per fare il GANNT
* Microsoft Word, per documentare
* Visual Studio Code, per programmare e sviluppare

### Hardware

PC della scuola

* Processore: 13th Gen Intel(R) Core(TM) i7-13700
* RAM: 32GB
* Scheda grafica: NVIDIA T400

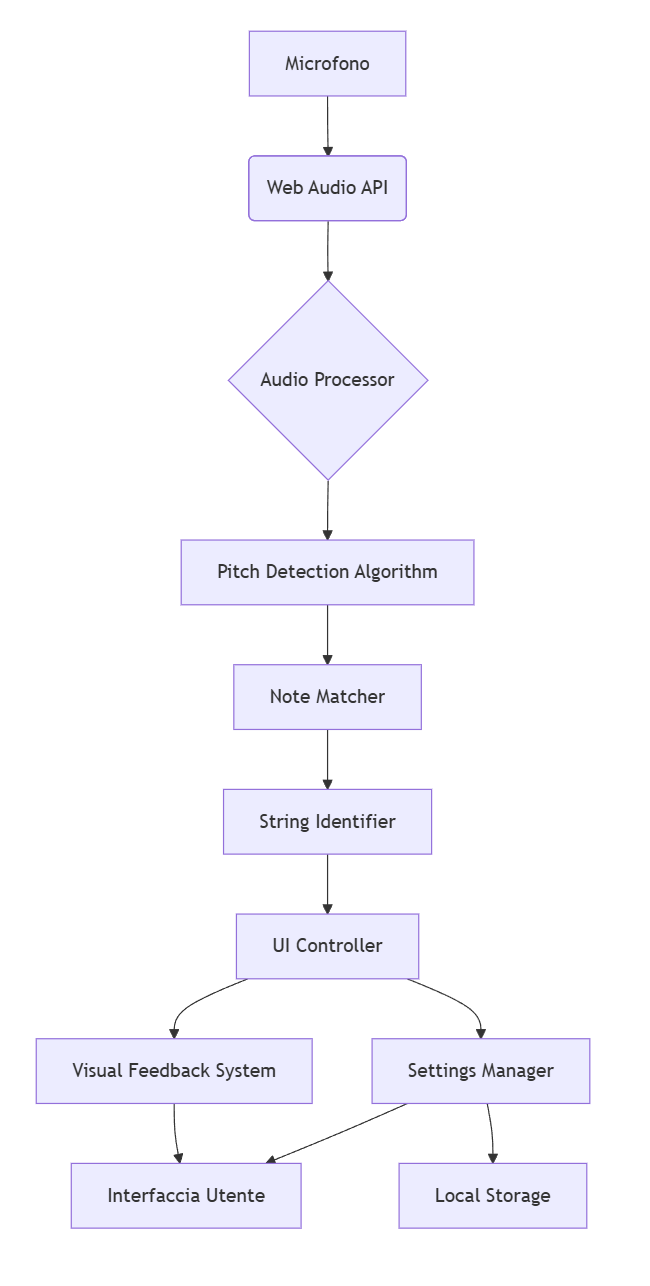
MacBook Air 2020 (personale)

* Apple Silicon M1
* 8GB di memoria unificata
* Microfono integrato

Strumento da accordare, chitarra ed ukulele personali.

# Progettazione

## Design dell’architettura del sistema



## Design delle interfacce

## 

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

# 

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card detailsExecute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …