

Report

Diego Roberto Ammirabile 0001021216
diego.ammirabile@studio.unibo.it

Febbraio 2025

Indice

1	Panoramica del progetto	1
1.1	Obiettivi	1
2	Dettagli di implementazione	2
2.1	Struttura del codice	2
2.2	Viste	3
2.3	Componenti	3
2.3.1	Componente Audio e sotto-componenti	4
2.4	utility	4
3	Interfaccia utente	4
3.1	Login e registrazione	4
3.2	Registrazione audio	5
3.3	Mappa	6
3.4	Audio salvati	7
4	Conclusioni	8

1 Panoramica del progetto

1.1 Obiettivi

Il progetto ha implementato le seguente funzionalità:

- registrazione e login dell'utente
- salvataggio e gestione della scadenza del token di autenticazione
- registrazione di audio
- memorizzazione di audio registrati su database locale SQLite
- visualizzazione di audio registrati
- caricamento di audio registrati sull'API
- eliminazione di audio registrati dal database locale e dall'API
- geolocalizzazione dell'utente
- visualizzazione di audio sulla mappa con relative informazioni
- visualizzazione di informazioni dell'audio salvato
- caricamento posticipato degli audio

- implementazione mostra/nascondi audio sulla mappa

Da specifiche erano richieste ulteriori funzionalità, tuttavia non sono state implementate a causa di problemi tecnici e di tempo. Questo argomento verrà trattato più avanti nelle conclusioni.

2 Dettagli di implementazione

Il progetto è stato realizzato attraverso grazie al framework **Ionic** combinato con il framework **Vue**, quindi l'applicazione è stata sviluppata utilizzando MVVM come design pattern. Quando possibile è stato utilizzato il **Capacitor** per accedere alle funzionalità native dei dispositivi mobili, questo non è sempre stato possibile a causa di limitazioni del framework. Per avere una migliore gestione dei tipi è stato utilizzato **TypeScript**. Il progetto è stato inizializzato con **Vite**.

Ionic è stato scelto poiché permette di “riciclare” le conoscenze di programmazione web e la familiarità con i framework javascript, inoltre permette di generare applicazioni per più piattaforme con un unico codice sorgente, caratteristica che però non è stata sfruttata.

Una comodità da non sottovalutare è la leggerezza dell'ambiente di sviluppo comparata ad *Android Studio* o *XCode*, inoltre la possibilità di utilizzare un browser per testare l'applicazione è un vantaggio non da poco.

Esistono in oltre delle librerie di componenti già pronti, che permettono di velocizzare lo sviluppo e si possono sfruttare le librerie sviluppate per javascript/typescript. Tuttavia non sempre funzionano correttamente e spesso è necessario apportare modifiche.

2.1 Struttura del codice

Il codice è stato organizzato in componenti Vue, in modo da rendere il codice più modulare e manutenibile. Ogni componente è stato progettato per essere il più possibile indipendente dagli altri, in modo da poter essere riutilizzato in altri contesti. Esistono tuttavia componenti che non trovano applicazione in altri contesti se non quello del sovra-componente, tali componenti però gestiscono parti di logica differente da quella del sovra-componente, per cui è stato ritenuto utile mantenerli separati. Il filesystem ha quindi la seguente struttura:

- **android/** contiene i file necessari per la compilazione dell'applicazione per Android
- **dist/** contiene i file compilati
- **node_modules/** contiene le dipendenze del progetto
- **public/** contiene i file statici
- **src/** contiene il codice sorgente
 - **components/** contiene i componenti Vue
 - **router/** contiene il router
 - **theme/** contiene i file di stile
 - **utils/** contiene le utility
 - **views/** contiene le viste
 - **App.vue** è il file principale
 - **interfaces.ts** contiene le interfacce
 - **main.ts** è il file principale
 - **variables.json** contiene le variabili, di fatto solo l'URL dell'API
 - **vite-env.d.ts** contiene le variabili di ambiente
- **capacitor.config.ts** contiene la configurazione di Capacitor
- **cypress.config.ts** contiene la configurazione di Cypress

- `index.html` è il file principale
- `ionic.config.json` contiene la configurazione di Ionic
- `LICENSE` contiene la licenza
- `package-lock.json` contiene le dipendenze con le versioni esatte
- `package.json` contiene le dipendenze
- `ProjectsLAM2024.pdf` contiene il progettato
- `README.md` contiene le informazioni sul progettato
- `tsconfig.json` contiene la configurazione di TypeScript
- `tsconfig.node.json` contiene la configurazione di TypeScript per Node
- `vite.config.ts` contiene la configurazione di Vite

2.2 Viste

Nella cartella `src/views/` sono presenti le viste dell'applicazione, queste sono state progettate per gestire una parte della logica dell'applicazione, in modo da rendere il codice più manutenibile e modulare. Le viste vengono caricate tramite il router che si trova nel file `src/router/index.ts`. A seconda delle viste è stata scelto se caricare i componenti in modo sincrono o asincrono, in modo da velocizzare il caricamento dell'applicazione. Per esempio la vista di login è caricata in modo sincrono nel seguente modo:

```
{
  path: '/login',
  name: 'login',
  component: LoginPage
}
```

mentre la vista della mappa, più impegnativa dal punto di vista delle risorse, è caricata in modo asincrono nel seguente modo:

```
{
  path: 'tab2',
  component: () => import('@views/MapPage.vue')
},
```

2.3 Componenti

Nella cartella `src/components/` sono presenti i componenti dell'applicazione, questi sono stati progettati per evitare ripetizione di codice e per rispecchiare un'unità logica dell'applicazione. La struttura dei componenti è la classica struttura di un componente Vue:

```
<template>
<!-- HTML -->
</template>
<script setup lang="ts">
// TypeScript
</script>
<style scoped>
/* CSS */
</style>
```

2.3.1 Componente Audio e sotto-componenti

Il componente Audio è il componente più complesso dell'applicazione, è stato progettato per gestire la registrazione e la visualizzazione degli audio. Questo componente è composto da diversi sotto-componenti, che gestiscono parti della logica differenti. La sua view ha la seguente struttura:

```
<template>
  <div id="card" v-if="visible">
    <div id="base-info">
      <IconButton fill="clear" @click="playFunction">
        <IonIcon :icon="playIcon" :disabled="props.audio.audioBase64===''"></IonIcon>
      </IconButton>
      <AudioTimestamp :duration="props.audio.duration" ref="timestamp"/>
      <AudioPreview :audio="audio" v-if="!props.audio.metadata" @delete="deleted"
        @upload="(uploadedAudio) => {props.audio.metadata = uploadedAudio.metadata}"/>
      <AudioUploaded :audio="audio" v-else @delete="deleted" @showMoreInfo=
        "toggleMoreInfo" />
    </div>
    <MetadataInfo :metadata="metadata" v-if="moreInfoVisible && metadata"/>
  </div>
</template>
```

tra i sotto-componenti vanno annoverati `AudioTimestamp` che si occupa di mostrare lo scorrere del tempo durante l'ascolto (è anche utilizzato per visualizzare lo scorrere del tempo durante la registrazione), `AudioPreview` che visualizza la parte dell'interfaccia visualizzata per gli audio non caricati e `AudioUploaded` che visualizza la parte dell'interfaccia visualizzata per gli audio caricati.

`MetadataInfo` è un componente che visualizza le informazioni dell'audio, è utilizzato sia per visualizzare le informazioni degli audio registrati dall'utente, sia per visualizzare le informazioni degli audio registrati dagli altri utenti. Nel componente è presente un bottone per mostrare/nascondere le informazioni dell'audio.

2.4 utility

Nella cartella `src/utils/` sono presenti le utility dell'applicazione, queste sono state progettate per evitare ripetizione di codice e per rispecchiare un'unità logica dell'applicazione, in una certa misura ogni file corrisponde a un model, quando questo è utile a più componenti. Questi sono in particolare i file:

- `audio_processing.ts` contiene le funzioni per la conversione dei codec audio tramite `FFmpeg` implementato in `WebAssembly` ¹
- `geolocation.ts` contiene le funzioni per la geolocalizzazione, implementate tramite `Capacitor` ²
- `hash.ts` contiene le funzioni per l'hashing delle password, implementate tramite `typescript`
- `requests.ts` contiene le funzioni per le richieste all'API, implementate tramite `Capacitor`
- `storage.ts` contiene le funzioni per la gestione del database locale, implementate tramite `Capacitor` ³

3 Interfaccia utente

3.1 Login e registrazione

La prima schermata che si presenta all'utente è quella di login, questa pagina contiene anche il collegamento alla pagina di registrazione dell'utente. È possibile salvare le credenziali dell'utente per evitare di doverle inserire ogni volta.⁴ Questa schermata è mostrata in figura 1. La schermata di registrazione è

¹FFmpeg è implementato in WebAssembly da <https://www.npmjs.com/package/@ffmpeg/ffmpeg>

²La geolocalizzazione è implementata tramite la libreria di Capacitor <https://capacitorjs.com/docs/apis/geolocation>


³Il database locale è implementato tramite la libreria della comunità di Capacitor <https://npmjs.com/package/@capacitor-community/sqlite>

⁴Le credenziali sono salvate tramite <https://capacitorjs.com/docs/apis/preferences>

mostrata in figura 2.

Login

Username angolo180

Password ●●●●●●●● 


☐ Remeber me

[LOGIN](#)

Don't have an account? Singup [here](#)

Sign Up

Username

Password 

[SIGN UP](#)

Already have an account? [Login](#)

Figura 1: Schermata di login

Figura 2: Schermata di registrazione

3.2 Registrazione audio

La schermata di registrazione audio è mostrata in figura 3. Questa schermata permette di registrare audio e di salvarlo nel database locale. È possibile ascoltare l'audio registrato prima di caricarlo sull'API. La schermata di registrazione audio con audio registrato è mostrata in figura 4, mentre la schermata di registrazione dopo che l'audio è stato caricato sull'API è mostrata in figura 5.

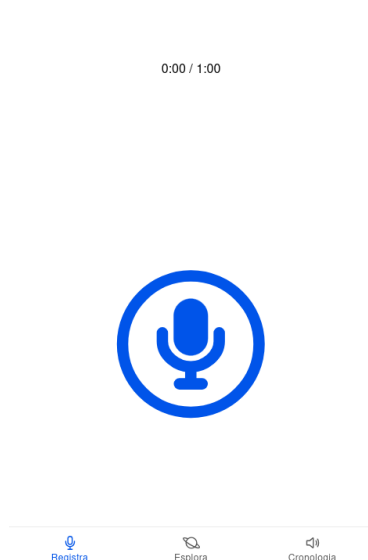


Figura 3: Schermata di registrazione audio

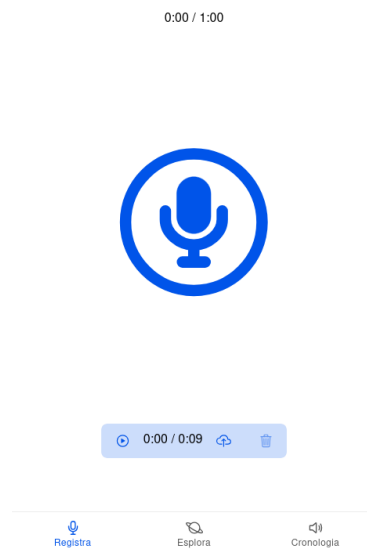


Figura 4: Schermata di registrazione audio con audio registrato

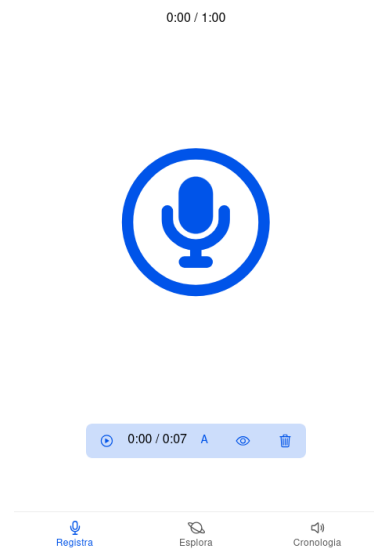


Figura 5: Schermata di registrazione audio con audio caricato

3.3 Mappa

La schermata della mappa con i pin è mostrata in figura 6. Questa schermata permette di visualizzare gli audio registrati sulla mappa. La mappa è stata realizzata tramite **Leaflet** ⁵, non si trattava di un componente nativo, ma ciononostante è stato preferito a **Google Maps**, che dispone di una implementazione nativa ⁶ l'interfaccia è più pulita, la documentazione è più chiara ed è più facilmente personalizzabile.

È possibile visualizzare le informazioni degli audio registrati cliccando sul pin, come mostrato in figura 7. Le informazioni mostrate sono ricavate dall'API e il componente utilizzato per visualizzare le informazioni lo stesso usato per la visualizzazione delle informazioni degli audio del utente caricato, mostrata in figura 10.

È possibile visualizzare più informazioni sul file audio cliccando sui menù a tendina dell'audio registrato, come mostrato in figura 8.

⁵Leaflet è una libreria per la creazione di mappe interattive <https://leafletjs.com/>

⁶Google Maps dispone di una implementazione nativa ufficiale di Capacitor; link alla documentazione: <https://capacitorjs.com/docs/apis/google-maps>

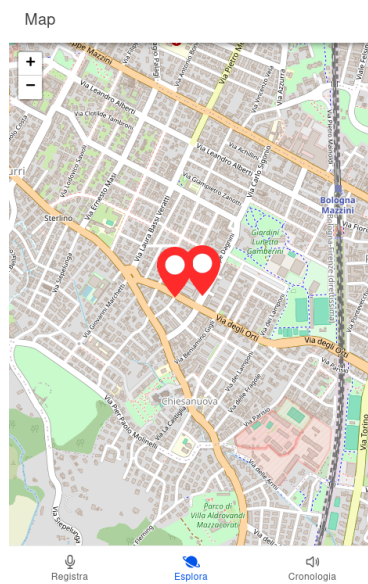


Figura 6: Schermata della mappa con i pin

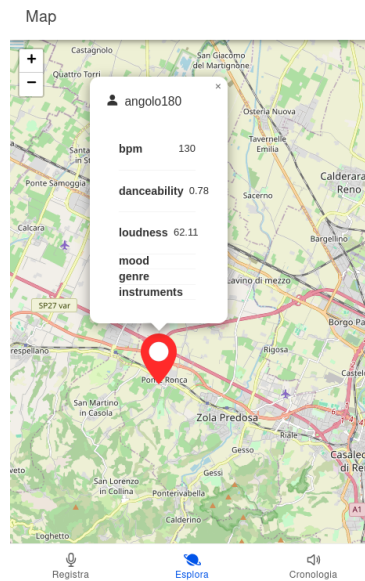


Figura 7: Schermata della mappa con le informazioni dell'audio

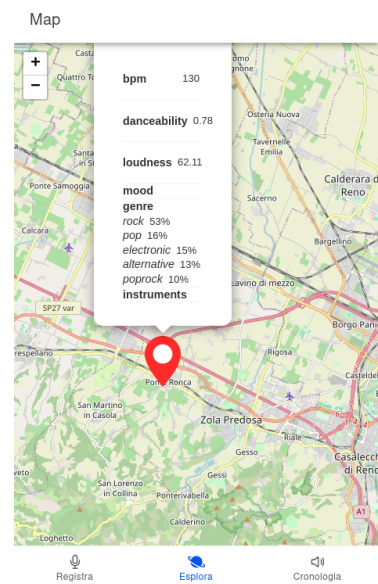


Figura 8: Schermata della mappa con un sotto-menu aperto

3.4 Audio salvati

La schermata degli audio salvati è mostrata in figura 9. Questa schermata permette di visualizzare gli audio registrati dall'utente. È possibile visualizzare le informazioni degli audio registrati cliccando sulla lettera "A", come mostrato in figura 10. Inoltre gli audio salvati sono riascoltabili, cancellabili e, qualora non siano stati caricati sull'API, è possibile caricarli.

Infine è possibile nascondere gli audio agli altri utenti, tramite l'icona dell'occhio.

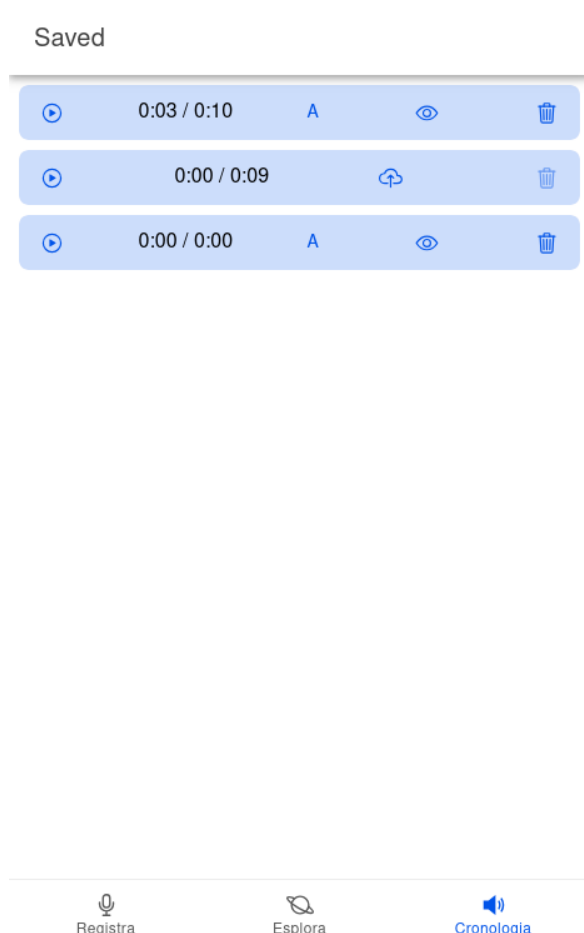


Figura 9: Schermata degli audio salvati

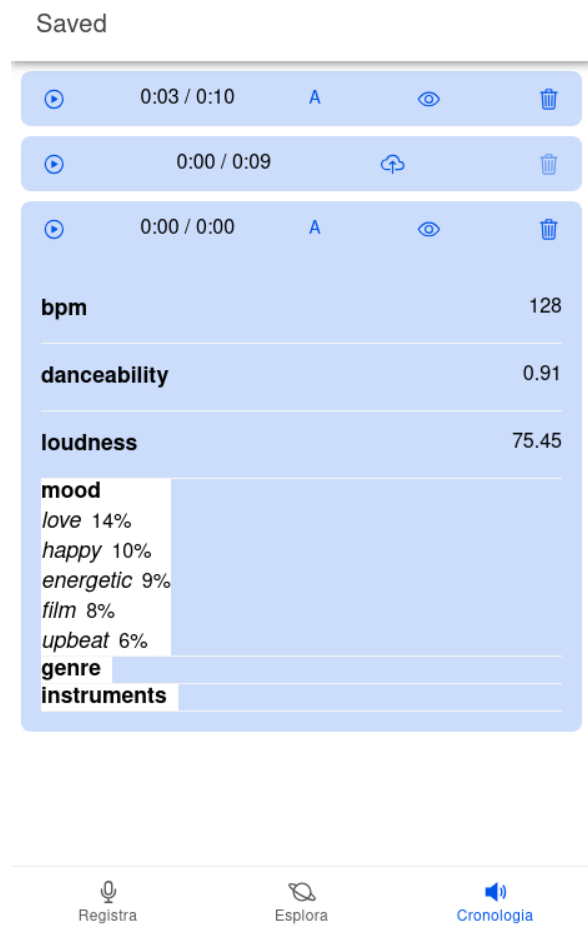


Figura 10: Schermata degli audio salvati con le informazioni

Esiste infine il caso in cui un audio sia stato caricato tramite l'account, e quindi sia visibile agli altri utenti, ma non sia stato caricato dal dispositivo, in questo caso l'audio è mostrato tra i salvati, ma non è possibile riascoltarlo pur rimanendo possibile cancellarlo o nascondere alla visualizzazione degli altri utenti.

4 Conclusioni

Il progetto è stato molto interessante e ha permesso di approfondire le conoscenze di programmazione mobile e di utilizzo di framework moderni. Tuttavia è stato molto impegnativo e non è stato possibile implementare tutte le funzionalità richieste.

Durante lo sviluppo sono stati riscontrati diversi problemi, che mi hanno portato a rivalutare Ionic + Capacitor da strumenti per sviluppo mobile per programmatori web a strumenti per sviluppo mobile per programmatori che intendono sviluppare applicazioni per più piattaforme. Questo perché il processo di porting da web a mobile è stato molto più complicato del previsto, spesso le funzionalità non funzionavano come previsto è stato necessario apportare modifiche al codice. E questi sforzi sono giustificabili solo nel caso in cui si voglia sviluppare un'applicazione per più piattaforme.

Ho notato in oltre come la documentazione di Capacitor sia molto scarna e spesso non aggiornata, il che è dovuto soprattutto alla velocità con cui il framework cambia, per esempio questo progetto è stato avviato utilizzando Capacitor 6.0.0, in quel momento la versione era molto nuova e molte librerie sviluppate dalla comunità non erano ancora state aggiornate; al momento della consegna la versione attuale è la 7.0.1.

Questa velocità di cambiamento è un problema per chi sviluppa applicazioni poiché rende non più compatibili le librerie sviluppate per versioni precedenti e quindi una libreria con 2 o 3 anni di vita può essere considerata obsoleta.

Infine ho riscontrato come l'ambiente nativo messo a disposizione da Capacitor talvolta non sia sufficiente, per esempio è prevista una libreria per la registrazione video, ma non è prevista una libreria per la registrazione audio, per cui è stato necessario utilizzare una libreria di terze parti, non inclusa nemmeno nelle librerie della comunità di Capacitor.⁷

Mi ritengo comunque soddisfatto del lavoro svolto perché benché lo sviluppo sia stato più impegnativo e lento di quanto avevo preventivato perché tutta la sua durata dello sviluppo ho continuato a imparare cose nuove e a migliorare le mie competenze. Con una certa fiducia posso dire che con più tempo e più esperienza sarei in grado di implementare tutte le funzionalità richieste, tuttavia se potessi tornare indietro non sceglierei Ionic + Capacitor per sviluppare un'applicazione mobile.

⁷Le della comunità di Capacitor sono disponibili all'indirizzo <https://npmjs.com/~capacitor-community>