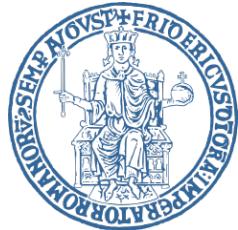


UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II



Corso di Ingegneria del Software

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE
DELL'INFORMAZIONE



Enjoy music, enjoy the lab

Candidati:

Fabrizio Di Giovanni - N8600-
Lucia Brando - N86003382
Dario Morace - N8600-
ID GRUPPO: - - - -

Docenti:

Prof. Sergio Di Martino
Prof. Francesco Cutugno

ANNO ACCADEMICO 2023/2024

Indice

1	Introduzione	4
1.1	Che cos'è SoundLab	4
1.2	Cosa offre Soundlab	5
1.3	Tecnologie utilizzate	5
2	Modello Funzionale	5
2.1	Requisiti dell'applicazione	6
2.1.1	Non funzionali	7
2.1.2	Dominio	7
2.2	Modellazione degli Use Case	8
2.2.1	Caso generale	8
2.2.2	Use Case azioni playlist	9
2.2.3	Use Case aggiunta playlist ai preferiti	10
2.2.4	Use Case aggiunta brano dalla playlist	11
2.2.5	Use Case azioni analitiche	11
2.3	Tabelle di Cockburn	13
2.4	UX Design	19
2.4.1	Mockup	22
2.4.2	Mockup Visualizzazione analitiche	24
2.4.3	Mockup Aggiunta traccia musicale ad una playlist	32
2.5	Presentazione dell'idea progettuale	37
2.5.1	Perchè abbiamo deciso di lavorare con la musica?	37
2.5.2	Analisi delle funzionalità	37
2.5.3	Stato di sviluppo delle funzionalità	38
2.6	Individuazione del target di utenti	41
2.6.1	Definizione delle Personas	41
2.7	Valutazione dell'usabilità a priori	46
2.7.1	Tabelle di valutazione e tecniche utilizzate	46
2.8	Prototipazione funzionale via statechart dell'interfaccia grafica	50
2.9	Glossario	51
2.9.1	Termini	51
3	Modelli di dominio	52
3.1	Classi, oggetti e relazioni di analisi	52
3.1.1	Classi ed entità	52
3.1.2	Class diagram delle funzionalità	53
3.2	Sequence diagram	57
3.2.1	Sequence Diagram Aggiungi playlist	57
3.2.2	Sequence Diagram Elimina profilo	58
3.3	Activity diagram	58
3.3.1	Activity diagram	59

4 Design di sistema	64
4.1 Analisi architetturale	64
4.1.1 Descrizione architettura cloud	64
4.1.2 Il server	66
4.1.3 Rest Api	67
4.1.4 Il client	69
4.1.5 Supporti	71
4.2 Class Diagram di Design	74
4.2.1 Classi Diagram Playlist	75
4.2.2 Class Diagram User	76
4.2.3 Class Diagram Library	77
4.2.4 Class Diagram Listening	78
4.3 Sequence Diagram di design	79
4.3.1 Sequence diagram Playlist	79
4.3.2 Sequence diagram User	81
4.4 Gerarchie funzionali	83
4.4.1 Gerarchie funzionali Login	83
4.4.2 Gerarchie funzionali Homepage	84
5 Codice xUnit	86
5.1 Test InsertOneListening()	87
5.2 Test RemoveSongFromPlaylist()	88
5.3 Test InsertOneListening	88
5.4 Test testRegistration()	89
5.5 Test ChangePassword() - WhiteBox	90
6 Valutazione dell'usabilità sul campo	92
6.1 Valutazione dell'applicativo	93
6.1.1 Metodo euristico	93
6.1.2 Metodo con monitoraggio ed analisi dei log	97
7 Conclusione e valutazione del team	98

1 Introduzione

Si vuole realizzare, dietro commissione della società SoftEngUniNA, una piattaforma per gli appassionati di musica. Nello specifico, la società ha richiesto la progettazione e implementazione di un applicativo mobile, oltre alla parziale verifica dei moduli necessari per il suo corretto funzionamento. Nasce così **SoundLab**, un nuovo spazio dedicato a tutti coloro che fanno della musica la colonna portante delle loro giornate.

1.1 Che cos'è SoundLab

SoundLab è un servizio dedicato agli appassionati di musica, offrendo una vasta selezione di brani e generi musicali per soddisfare i gusti di tutti gli utenti registrati. Gli utenti possono accedere al sistema per esplorare e ascoltare musica, creare e gestire le proprie playlist, visibili sui loro profili personali.

Inoltre, gli amministratori hanno accesso a strumenti di analisi avanzati, permettendo loro di visualizzare le statistiche relative ai brani e agli utenti presenti nel sistema, migliorando così l'esperienza complessiva degli utenti.



In conformità con i recenti standard previsti dall'Unione Europea, SoundLab si impegna all'assoluto rispetto della privacy e dei dati della propria utenza per assicurare un'esperienza di navigazione sicura; proprio perché abbiamo a cuore tale sicurezza, il nostro staff si impegna a offre un'assistenza a 360 gradi.

1.2 Cosa offre Soundlab

Per offrire un'esperienza ottimale, SoundLab si basa su un robusto *back-end* progettato per garantire prestazioni elevate e scalabilità in base alle esigenze dell'utenza. Questo sistema back-end avanzato assicura che l'applicazione possa gestire efficacemente un numero crescente di utenti e richieste, mantenendo sempre una risposta rapida e affidabile.

Dall'altro lato, Soundlab presenta un'interfaccia *front-end* semplice e moderna, progettata per rendere intuitiva la navigazione. Gli utenti possono facilmente cercare, esplorare e interagire con contenuti e funzionalità dell'applicazione, rendendo la loro esperienza unica e adattabile alle loro preferenze personali.

La combinazione di un back-end potente e un'interfaccia utente intuitiva permette a SoundLab di offrire un servizio eccezionale, migliorando la soddisfazione degli utenti e favorendo un'interazione più profonda con la piattaforma.

1.3 Tecnologie utilizzate

L'applicativo è sviluppato interamente su piattaforma *Android* utilizzando il linguaggio di programmazione *Object-Oriented*, in particolare *Java*.

Questa scelta tecnologica garantisce robustezza, flessibilità e una vasta compatibilità con dispositivi Android di diverse generazioni. Inoltre, l'applicazione è dotata di un sistema di logging avanzato che facilita il testing efficace e la risoluzione rapida dei problemi, migliorando la qualità complessiva del software e l'esperienza utente.

Per il back-end, SoundLab sfrutta tecnologie all'avanguardia, tra cui servizi di *public Cloud Computing* che permettono di massimizzare la scalabilità e le prestazioni del sistema. Questi servizi cloud offrono una piattaforma affidabile e sicura per la gestione dei dati, consentendo al sistema di adattarsi dinamicamente al crescente numero di utenti e alle richieste variabili. La combinazione di un'applicazione front-end potente e intuitiva con un back-end scalabile e performante garantisce un servizio eccellente e un'esperienza utente ottimale.

Nei capitoli successivi, approfondiremo ulteriormente il discorso sulle tecnologie utilizzate e forniremo, come richiesto da SoftEngUniNa, i documenti relativi al lavoro svolto per la realizzazione del progetto.

2 Modello Funzionale

In questa sezione andremo a descrivere il modello funzionale del software partendo da un'analisi dei casi d'uso assegnati per l'applicativo.

2.1 Requisiti dell'applicazione

Funzionalità	Descrizione
Ricerca delle tracce musicali e/o artista	Ricerca tramite nome una traccia musicale oppure un'artista all'interno della piattaforma.
Gestione playlist	L'utente ha la possibilità di creare, eliminare ed aggiungere/rimuovere una playlist dalle preferite.
Visualizzazione profilo	L'utente, in quanto iscritto, alla piattaforma può visualizzare il proprio profilo dove trova le playlist create e preferite.
Gestione profilo	L'utente, in quanto iscritto alla piattaforma, tramite i settings può gestire il proprio profilo. Si può: modificare l'username, modificare l'email, modificare la password, cancellare il proprio profilo o fare un semplice logout.
Gestione delle tracce musicali	L'utente può aggiungere e/o rimuovere una traccia musicale da una playlist.
Visualizzazione statistiche (solo per l'utente admin)	Tramite un'apposito bottone presente sul proprio profilo, l'admin può recuperare informazioni riguardo gli utenti, le fasce orarie in cui l'applicativo viene utilizzato e gli ascolti che effettuano.

2.1.1 Non funzionali

Funzionalità	Descrizione
Usabilità	L'applicazione deve presentarsi in modo semplice, così da permettere ad ogni tipologia di utente di fare uso di tutte le funzionalità che essa mette a disposizione.
Scalabilità	Il sistema deve potersi adattare ai cambiamenti del backend, in modo da garantire la manutenibilità nel tempo.
Password policy security	Le password verranno salvate con un controllo regex avanzato.
Prestazioni	Il sistema deve essere utilizzabile entro 3 secondi dall'avvio. Inoltre, non devono presentarsi rallentamenti che impediscono all'utente di effettuare operazioni.
Utilizzo di single-activity e multi-fragment	L'applicazione utilizza perfluidità e facile gestione il pattern di single-activity e multi-fragment in modo da dare precisi scopi alle activity che gestiscono fragment comuni.

2.1.2 Dominio

Dominio	Descrizione
General Data Protection Regulation	Il sistema deve essere conforme al Regolamento Generale sulla Protezione dei Dati (GDPR), incentrato sul trattamento dei dati personali e sulla privacy dell'utente
ISO/IEC 27018:2019	Il sistema deve essere conforme allo standard ISO/IEC 27018:2019[9], incentrato sulla protezione dei dati personali nel cloud.

2.2 Modellazione degli Use Case

Per la documentazione dei casi d'uso è stato utilizzato un software case tool, chiamato Visual Paradigm.

Per fini di visibilità e leggibilità dei singoli use case, si è deciso di modellare il diagramma in package.

2.2.1 Caso generale

Il seguente use case rappresenta l'insieme delle funzionalità del sistema assegnate dalla software house.

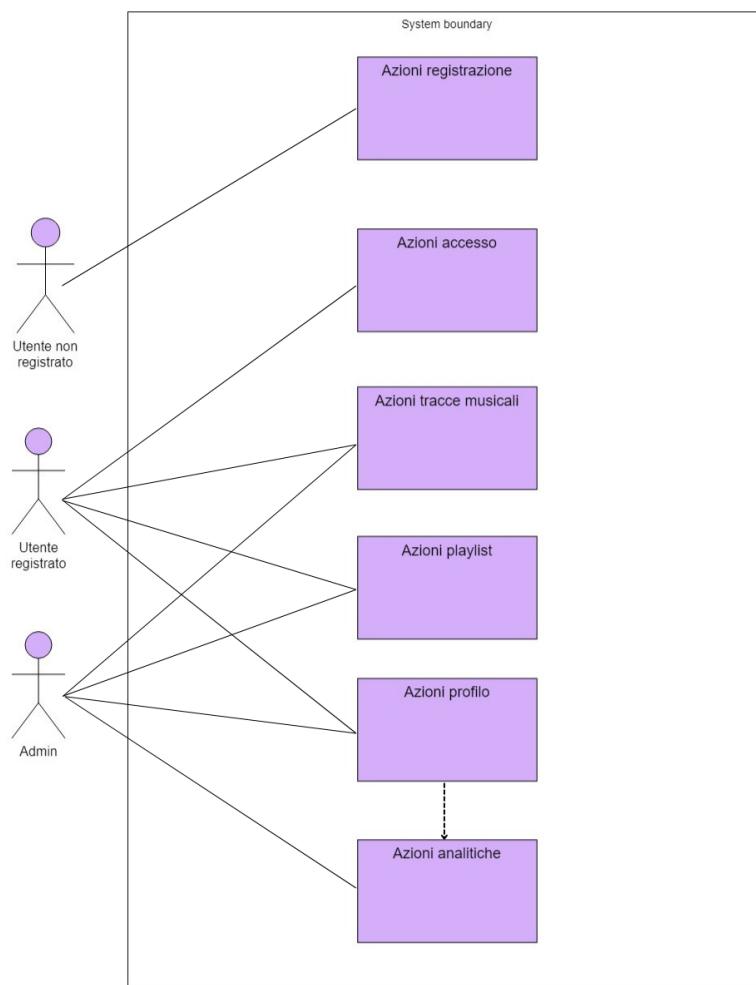


Figura 1: Use Case Diagram generale

2.2.2 Use Case azioni playlist

Nella seguente figura è descritto l'insieme delle azioni riguardanti le playlist. Si identificano le seguenti:

- Creare una nuova playlist
- Aggiungere un brano alla playlist
- Rimuovere un brano dalla playlist
- Rinominare una playlist
- Cambiare genere ad una playlist
- Aggiungere e/o rimuovere una playlist dalle preferite
- Eliminare una playlist

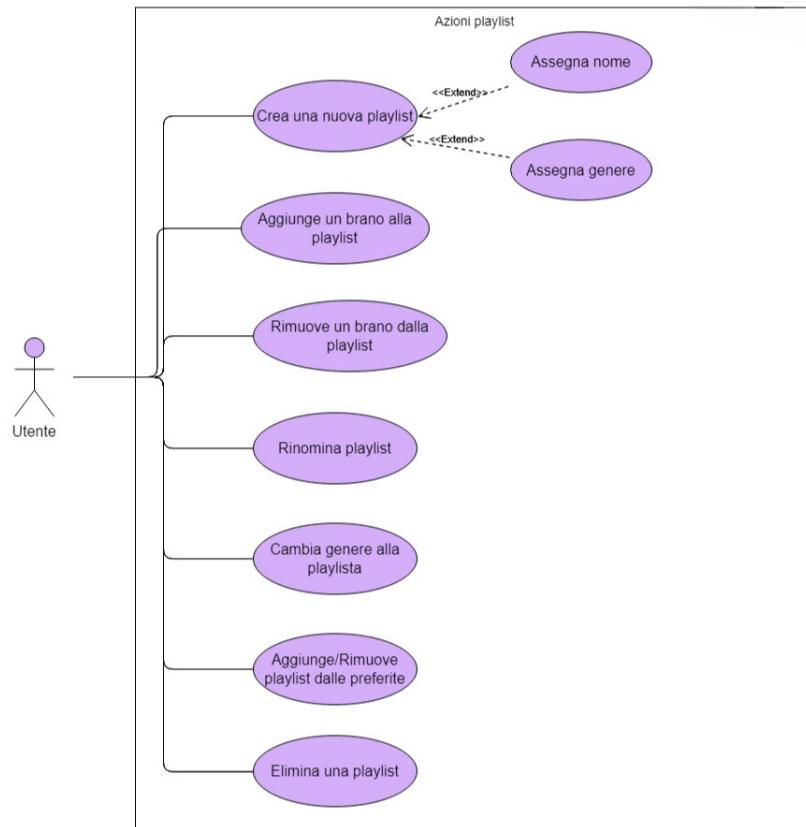


Figura 2: Use Case Diagram azioni playlist

2.2.3 Use Case aggiunta playlist ai preferiti

Nella seguente figura viene illustrato nel dettaglio il processo di aggiunta/rimozione di una playlist ai preferiti. L'utente, mediante l'utilizzo di un apposito toggle, può scegliere di includere o escludere tale playlist dall'elenco dei preferiti.

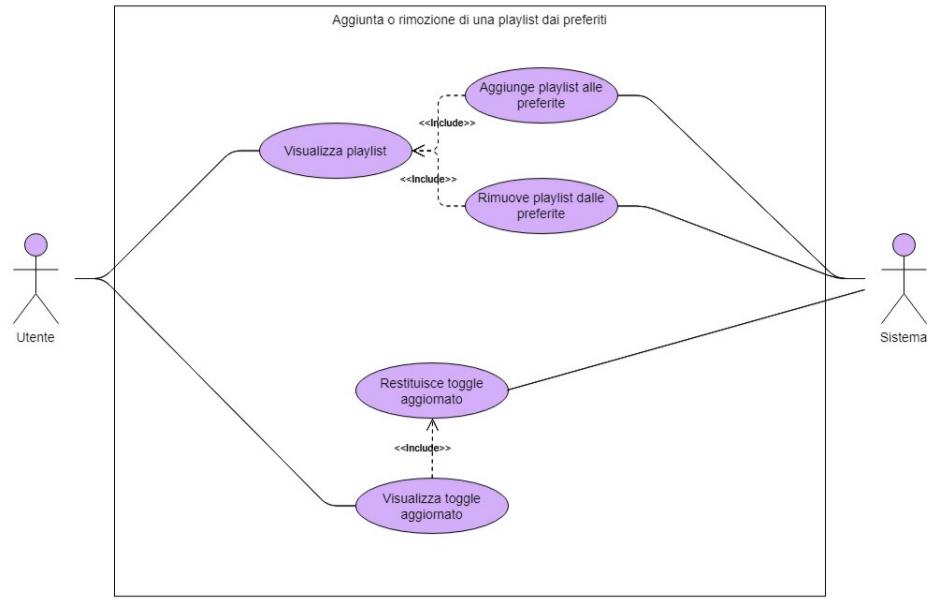


Figura 3: Use Case Diagram aggiunta/rimozione playlist dai preferiti

2.2.4 Use Case aggiunta brano dalla playlist

Nella seguente figura si è scelto di mostrare nel dettaglio l'aggiunta di una traccia musicale da una playlist.

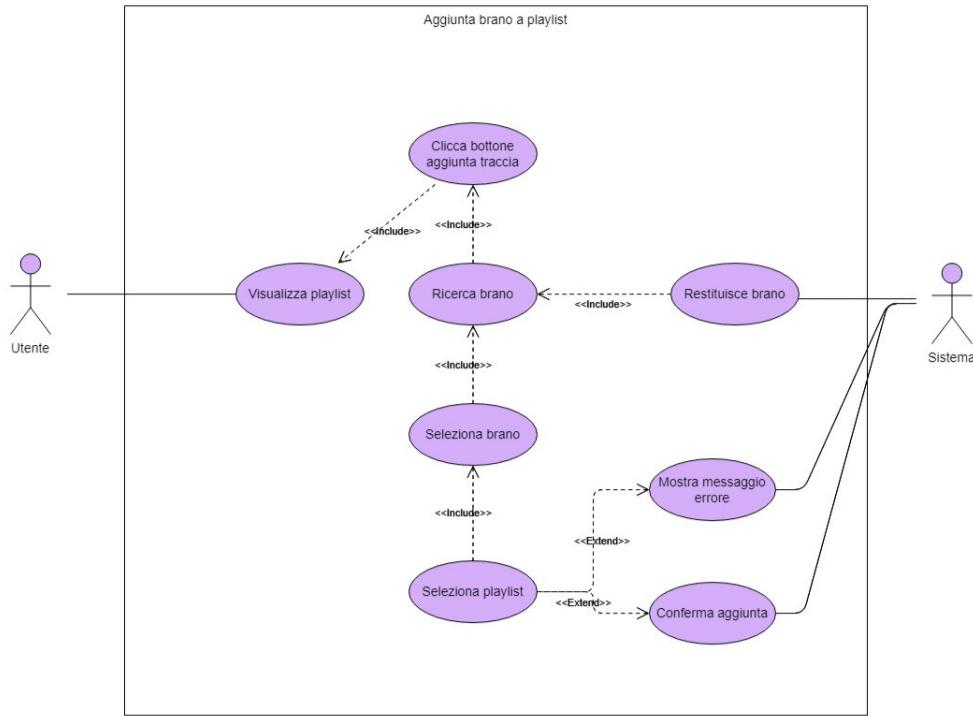


Figura 4: Use Case Diagram aggiunta traccia musicale ad una playlist

2.2.5 Use Case azioni analitiche

Il seguente use case rappresenta l'insieme di azioni che solamente l'utente admin può compiere. Le azioni in questione riguardano le analitiche:

- Se l'admin ricerca un utente, egli visualizzerà *il numero di ascolti compiuti e la fascia oraria in cui ha utilizzato di più la piattaforma*.
- Se l'admin ricerca una traccia musicale, egli visualizzerà *la tipologia della canzone, l'artista e il numero di ascolti totali del brano*.

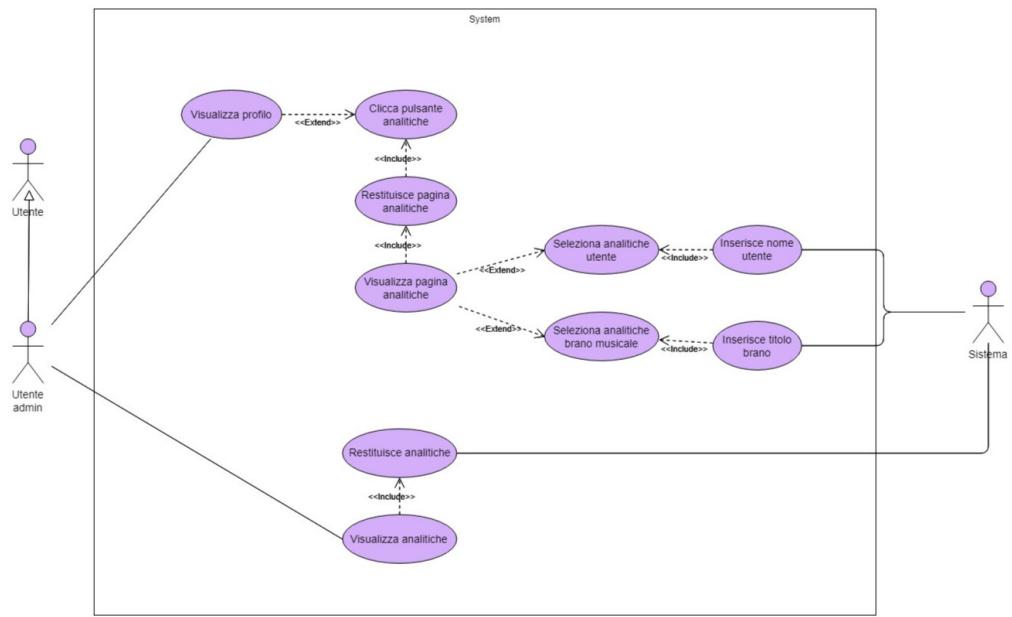


Figura 5: Use Case Diagram azioni analitiche

2.3 Tabelle di Cockburn

Le tabelle di Cockburn, che devono il loro nome all'informatico Alistair Cockburn, sono un formalismo di rappresentazione di casi d'uso; la politica adotta la rappresentazione di un *main* scenario nella quale uno o più attori interagiscono tra loro attraverso l'invocazione di *trigger* e descrivendo gli eventi (sottoforma tabulare).

Abbiamo deciso di rappresentare i seguenti Use Case:

- Aggiunta canzone ad una playlist
- Azioni analitiche (ricerca e visualizzazione)

Use Case #01	Aggiunta traccia musicale ad una playlist		
Goal in Context	L'utente deve riuscire ad aggiungere una traccia musicale ad una playlist		
Preconditions	L'utente deve essere registrato e deve avere sul proprio profilo almeno una playlist		
Success End Condition	L'utente è riuscito ad inserire una traccia musicale all'interno della playlist		
Failed End Condition	L'utente non è riuscito ad inserire una traccia musicale all'interno della playlist		
Actors	Utente, Sistema		
Trigger	Dopo aver visualizzato la playlist, l'utente clicca sul pulsante "Aggiungi una canzone"		
Description	Step	User Action	System Action
	1	Visualizza la playlist	
	2		Mostra la schermata "PlaylistFragment"
	3	Clicca sul pulsante "Aggiungi una canzone"	
	4		Mostra la schermata "SearchFragment"
	5	Scrive il titolo della canzone che vuole aggiungere alla playlist	
	6		Mostra la canzone ricercata e/o suoi simili
	7	Seleziona la canzone desiderata	
	8	Seleziona la playlist in cui vuole aggiungere una canzone	

	<p>9</p> <p>Se non ci sono problemi, permette l'inserimento della canzone all'interno della playlist</p>									
	<p>10</p> <p>Restituisce un messaggio di riuscita</p>									
Extension	<table border="1" data-bbox="714 555 1462 1039"> <thead> <tr> <th data-bbox="714 555 910 629">Step</th><th data-bbox="910 555 1171 629">User Action</th><th data-bbox="1171 555 1462 629">System Action</th></tr> </thead> <tbody> <tr> <td data-bbox="714 629 910 925">11.a</td><td data-bbox="910 629 1171 925"></td><td data-bbox="1171 629 1462 925">Mostra un pop-up d'errore di inserimento traccia già esistente nella playlist</td></tr> <tr> <td data-bbox="714 925 910 1039"></td><td data-bbox="910 925 1171 1039"></td><td data-bbox="1171 925 1462 1039">Torna al punto 4 del main scenario</td></tr> </tbody> </table>	Step	User Action	System Action	11.a		Mostra un pop-up d'errore di inserimento traccia già esistente nella playlist			Torna al punto 4 del main scenario
Step	User Action	System Action								
11.a		Mostra un pop-up d'errore di inserimento traccia già esistente nella playlist								
		Torna al punto 4 del main scenario								

Use Case #02	Visualizzazione analitiche		
Goal in Context	L'admin deve riuscire a visualizzare le analitiche ricercate		
Preconditions	L'utente deve essere registrato come admin , così da visualizzare il bottone apposito per le analitiche		
Success End Condition	L'admin è riuscito a visualizzare le analitiche		
Failed End Condition	L'admin non è riuscito a visualizzare le analitiche		
Actors	Admin, Sistema		
Trigger	Dopo aver visualizzato il profilo, l'admin clicca sul bottone delle analitiche, compilando i campi richiesti ed avviando la ricerca		
Description	Step	User Action	System Action
	1	Visualizza il profilo	
	2	Clicca sul pulsante dedicato alle analitiche	
	3		Mostra la schermata "AnaliticheFragment"
	4	Sceglie dal radiobutton se cercare le analitiche riguardanti un utente oppure ad una traccia musicale	
	5	Inserisce il nome dell'utente oppure della traccia musicale in questione	
	6	Clicca sul tasto di ricerca	
	7		Il sistema elabora la richiesta

	8	Se l'admin ha cercato un utente, il sistema visualizza il numero di ascolti compiuti e la fascia oraria in cui ha utilizzato di più la piattaforma
	9	Se l'admin ha cercato una traccia musicale, il sistema visualizza la tipologia della canzone, l'artista e il numero di ascolti totale del brano
	10	Visualizza le informazioni richieste
Subvariant 1	Step	User action
	11.a	Clicca sul pulsante per tornare indietro
		Mostra la schermata "ProfileFragment"
Subvariant 2	Step	User action
	12.b	Non compila tutti i campi necessari per la ricerca
		Restituisce un messaggio d'errore
Subvariant 3	Step	User action
	13.c	Seleziona dal radiobutton "Utente" ma scrive il nome di una traccia musicale
		Restituisce messaggio d'errore
Subvariant 4	Step	User action
	14.d	Seleziona il radiobutton "Traccia" ma scrive l'username di un utente

			Restituisce messaggio d'errore
Subvariant 5	Step	User action	System action
	15.e	Seleziona uno dei radiobotton ma inserisce un nome che non è presente sulla piattaforma	Restituisce messaggio d'errore

2.4 UX Design

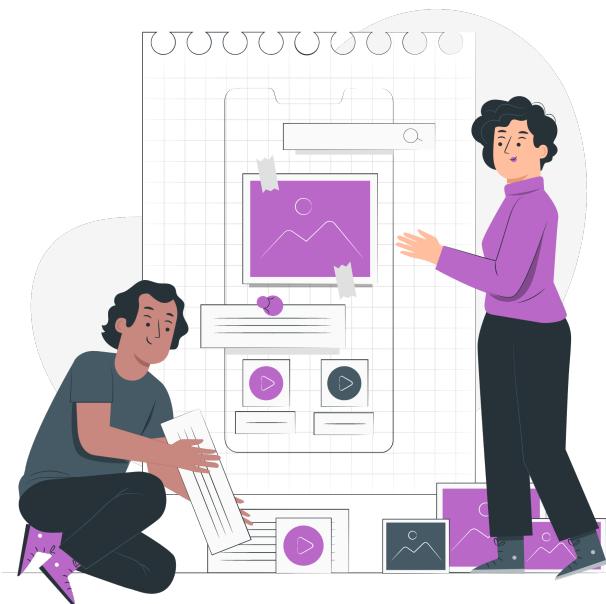
Il team ha approfondito diversi studi in materia di design della *user experience* e implementato quelle che sono state ritenute le scelte migliori intermini di colori, affordances e estetica per assicurare all'utente la migliore esperienza possibile sull'applicativo.

In termini di realizzazione pratica della GUI, la scelta è ricaduta:

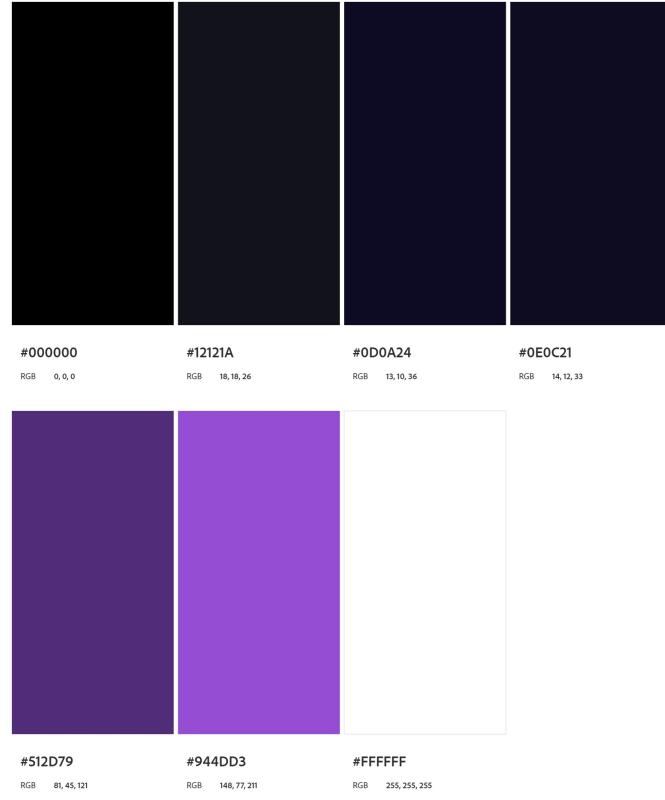
- Coerenza visiva con altre applicazioni simili per facilitarne l'uso dell'interfaccia.
- Impatto estetico degli elementi
- Adattabilità delle sue componenti con il dispositivo usato

Per garantire una buona esperienza utente, si concorda sull'importanza di applicare i principi della **Gestalt**, una corrente psicologia tedesca dei primi del Novecento che costruisce l'esperienza rispetto a diversi fenomeni.

Secondo la *Gestalt* gli elementi simili che costituiscono un'immagine o una composizione, vengono raggruppati tra loro e poi percepiti come un unico elemento.



Il colore e le forme hanno, dunque, un ruolo importantissimo nella creazione di un'interfaccia utente in quanto non solo migliora l'estetica ma è un veicolo di informazioni. La paletta colori che abbiamo deciso di utilizzare per il nostro applicativo è la seguente:



color.adobe.com



La scelta di tale paletta colori nasce dal nome stesso dell'applicativo:

- **Sound:** traduzione in inglese della parola "suono"
- **Lab:** diminutivo della parola inglese "Laboratory"

La scelta delle sfumature del color viola affiancate ai classici bianco e nero, rappresentano una forte connessione al mondo della musica ma anche al tema del laboratorio su cui verte l'applicativo.

- **Viola:** spesso associato alla creatività, all'innovazione e alla magia. Questi attributi sono fondamentali laddove si vogliano esplorare nuove idee, facendo della creatività il fulcro della composizione e dell'espressione artistica.
- **Nero:** colore elegante e sofisticato, che dona un aspetto raffinato e professionale. Inoltre, il nero può aiutare a mantenere il focus sugli elementi principali dell'interfaccia senza distrazioni.

- **Bianco:** offre un ottimo contrasto con i colori più scuri della paletta, migliorando la leggibilità del testo e la visibilità degli elementi interattivi.

La scelta di utilizzare **forme** più tondeggianti rispetto a quelle spigolose in termini di estetica ed usabilità può influenzare significativamente l'esperienza utente.

- Le forme tondeggianti sono spesso percepite come più morbide e accoglienti rispetto a quelle spigolose.
Questo può creare un'atmosfera più amichevole e invitante per l'utente, contribuendo a rendere l'app più piacevole e meno intimidatoria
- Le curve e le linee arrotondate sono caratteristiche comuni nel design moderno e contemporaneo.
Questi elementi possono dare all'app un aspetto più aggiornato e all'avanguardia, in linea con le tendenze attuali del design
- Le forme tondeggianti possono creare un senso di continuità e flusso visivo. La transizione tra gli elementi è più fluida, contribuendo a un'interfaccia più armoniosa e integrata
- Le forme tondeggianti possono trasmettere una sensazione di sicurezza e protezione, in quanto mancano degli angoli appuntiti che potrebbero sembrare minacciosi.
Questo può essere particolarmente importante per utenti con esigenze di accessibilità, rendendo l'app più inclusiva

Fin dal prototipo iniziale, i valori citati precedentemente sono stati sempre rispettati.

Bisogna specificare, però, che in fase di sviluppo molte scelte stilistiche sono state modificate per garantire una funzionalità maggiore dell'applicativo.

Per una visione completa dei mockup dell'applicazione, si consiglia di cliccare il seguente link:
[Figma - SoundLab](#)

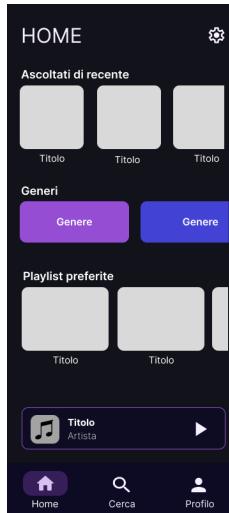


Figura 6: Prototipo iniziale della schermata di home



Figura 7: Prototipo iniziale della pagina di ricerca

2.4.1 Mockup

Il mockup è una rappresentazione visiva del prodotto o dell'idea, utilizzata per valutarne l'aspetto e l'organizzazione. È un modello statico e illustra l'aspetto e il funzionamento previsto di un prodotto, per esempio.

Esso comprende elementi, come per esempio l'aspetto grafico (i loghi, le immagini, i colori, le visualizzazioni della navigazione, ecc.), che saranno utilizzati nel design finale e nell'esperienza dell'utente. In sostanza, un mockup serve a mostrare agli stakeholder e agli utenti come potrebbe essere il progetto o il prodotto finito, offrendone un'idea visiva chiara e dettagliata del design e delle funzionalità.

I mockup sono stati fatti utilizzando come tool-case **Figma**. Di seguito, una prima versione del *mockup*¹.

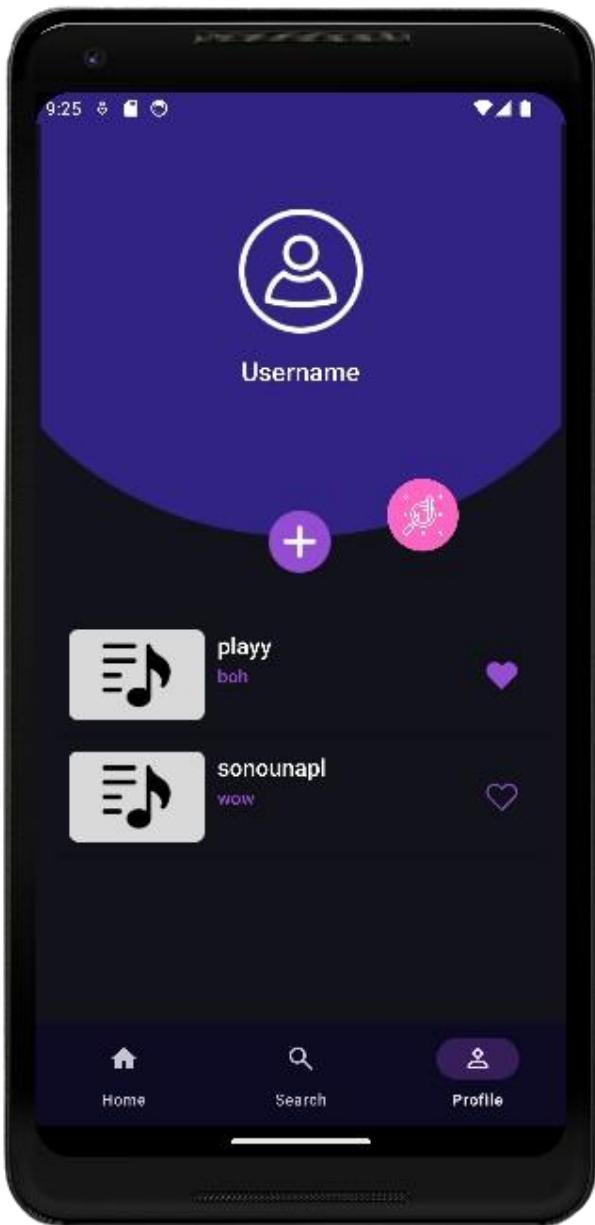
¹L'applicativo finale rispecchia in pieno i mockup iniziali. Tuttavia, bisogna specificare che in fase di sviluppo alcune scelte stilistiche sono state modificate per favorire la user experience



Verranno mostrati, di seguito, i *mockup finali* di due metodi scelti ovvero: **visualizzazione analitiche** e **aggiunta traccia musicale ad una playlist**.

2.4.2 Mockup Visualizzazione analitiche

NOTA: è stato ritenuto opportuno riportare anche un caso d'errore.











Errore 1: Sono stati inseriti due parametri diversi per la ricerca





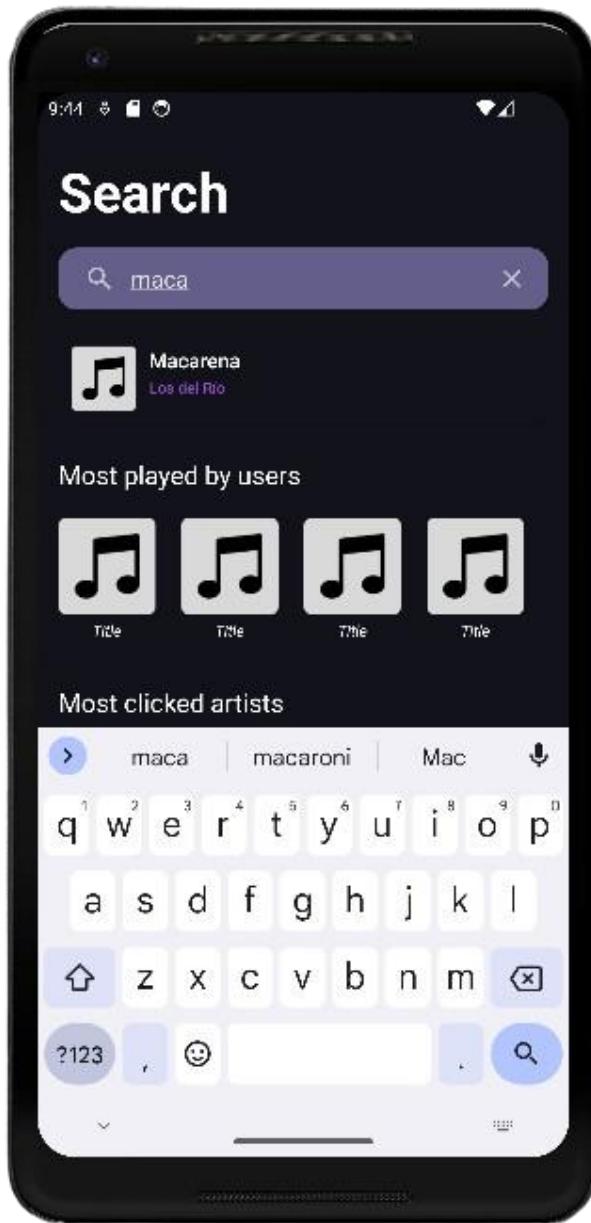
Errore 2: Non sono stati compilati tutti i campi necessari per la ricerca

2.4.3 Mockup Aggiunta traccia musicale ad una playlist

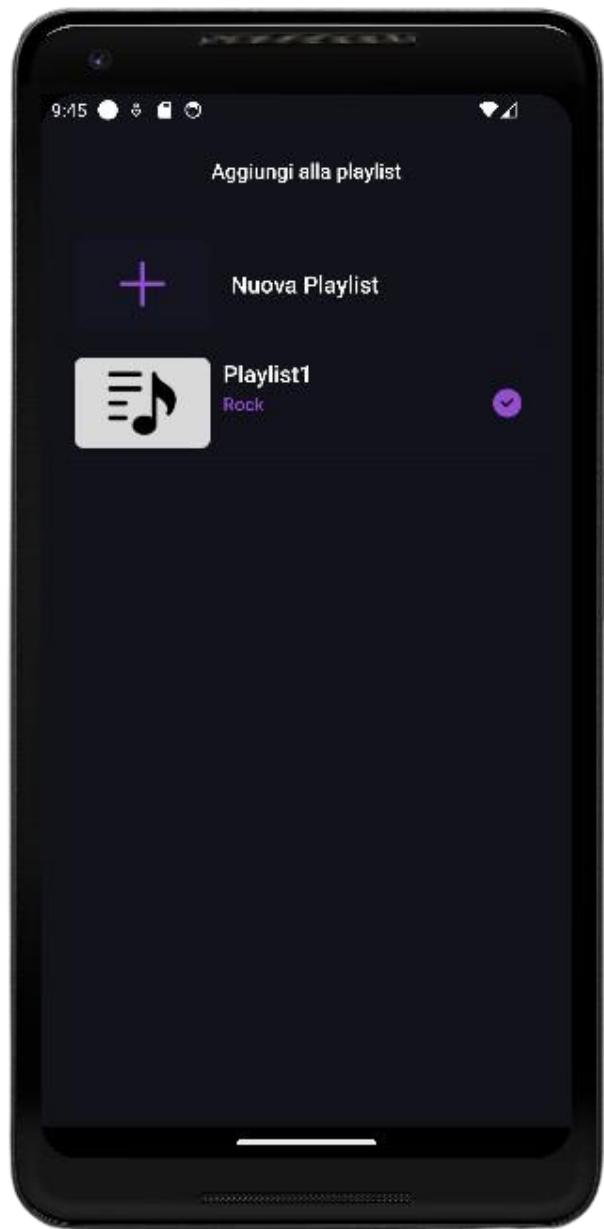




Mockup 2: Dopo aver cliccato su "Aggiungi una canzone" ricerchiamo la traccia desiderata



Mockup 3: Selezioniamo la canzone



Mockup 4: selezioniamo la playlist in cui aggiungerla

2.5 Presentazione dell'idea progettuale

Soundlab nasce dall'esigenza di creare una piattaforma di streaming musicale che offra condizioni più vantaggiose per gli artisti, garantendo una remunerazione più equa e trasparente. Inoltre, si propone di migliorare significativamente l'esperienza dell'utente attraverso un'interfaccia più intuitiva e funzionalità innovative che permettano una maggiore personalizzazione dei contenuti.

Soundlab mira anche a differenziarsi con un catalogo musicale unico, promuovendo così una maggiore diversità musicale. Infine, una priorità fondamentale è garantire una maggiore tutela della privacy e sicurezza dei dati degli utenti, rispondendo alle crescenti preoccupazioni riguardo alla gestione delle informazioni personali.

2.5.1 Perchè abbiamo deciso di lavorare con la musica?

Al giorno d'oggi, lavorare con la musica o attraverso piattaforme ad essa dedicate offre numerosi vantaggi.

Innanzitutto, la **crescente diffusione della tecnologia e l'accesso a Internet** hanno reso la musica più facilmente fruibile a livello globale, ampliando il pubblico potenziale e le opportunità di mercato per artisti e imprenditori.

Le piattaforme digitali consentono di **raggiungere rapidamente un vasto numero di ascoltatori**, superando le barriere geografiche e culturali.

Inoltre, l'analisi dei dati e gli algoritmi di raccomandazione permettono di **comprendere meglio le preferenze degli utenti**, offrendo esperienze musicali personalizzate e aumentando l'engagement. Questo può tradursi in un maggior numero di stream, vendite e visibilità per gli artisti.

Infine, lavorare con la musica attraverso piattaforme dedicate offre l'opportunità di **innovare costantemente**, sperimentando nuove tecnologie e format, come la realtà virtuale e aumentata, per arricchire l'esperienza dell'ascoltatore e mantenere un vantaggio competitivo in un settore in continua evoluzione.

2.5.2 Analisi delle funzionalità

- **Un utente può registrarsi:** Gli utenti possono creare un account personale fornendo le informazioni necessarie, come email e password, per accedere alle funzionalità avanzate dell'app.
- **Un utente registrato può creare la propria playlist:** Dopo la registrazione, gli utenti possono creare playlist personalizzate, dando loro un nome e aggiungendo i brani preferiti.
- **Un utente registrato può eliminare una playlist:** Gli utenti hanno la possibilità di eliminare le playlist che non desiderano più mantenere.
- **Un utente registrato può aggiungere o rimuovere un brano dalla playlist:** Gli utenti possono modificare le loro playlist aggiungendo nuovi brani o rimuovendo quelli esistenti in base ai propri gusti.
- **Un utente registrato può ricercare un brano musicale e/o una playlist:** Gli utenti possono utilizzare la funzione di ricerca per trovare specifici brani musicali o playlist, facilitando l'accesso ai contenuti desiderati.

- **L'admin può visualizzare le analitiche dell'applicativo:** Gli amministratori hanno accesso alle statistiche e alle analitiche dell'app, come la fascia oraria in cui l'applicazione è più utilizzata, e altre metriche utili per monitorare e migliorare il servizio.
- **Un utente, sia esso admin o no, può modificare le impostazioni del proprio profilo:** Tutti gli utenti, inclusi gli amministratori, possono personalizzare le impostazioni del proprio profilo, come aggiornare le informazioni personali, modificare la password, modificare l'email o cancellare il proprio profilo.

2.5.3 Stato di sviluppo delle funzionalità

Nelle seguenti tabelle è stato rappresentato lo stato di sviluppo delle funzionalità ad oggi:

	Login	Sign in	Logout
Idea progettuale			
Mockup e StateChart			
Prima implementazione			
Diagramma delle classi			
Testing			
Beta-testing		X	
Prodotto finito	X		X

	Crea playlist	Rinomina playlist	Elimina playlist	Cambia genere alla playlist	Aggiungi playlist ai preferiti
Idea progettuale					
Mockup e StateChart					
Prima implementazione					
Diagramma delle classi					
Testing					
Beta-testing					
Prodotto finito	X	X	X	X	X

	Aggiungi canzone a playlist	Rimuovi canzone a playlist	Ricerca canzone	Ricerca playlist	Riproduci brano
Idea progettuale				X	
Mockup e StateChart					
Prima implementazione					
Diagramma delle classi					
Testing					
Beta-testing	X	X			
Prodotto finito			X		X

	Modifica username	Modifica password	Modifica email	Elimina account
Idea progettuale				
Mockup e StateChart				
Prima implementazione				
Diagramma delle classi				
Testing				
Beta-testing				
Prodotto finito	X	X	X	X

Solo per l'utente admin	Ricerca e visualizza analitiche utente	Ricerca e visualizza analitiche brano musicale
Idea progettuale		
Mockup e StateChart		
Prima implementazione		
Diagramma delle classi		
Testing		
Beta-testing	X	X
Prodotto finito		

2.6 Individuazione del target di utenti

Il target di utenti che si può definire da una prima (e relativa) analisi dei casi d'uso sono ovviamente:

- Utenti appassionati di musica
- Utenti che utilizzano piattaforme digitali
- Utenti interessati alla personalizzazione della loro esperienza musicale

Dal periodo della pandemia, l'utilizzo delle piattaforme di streaming musicale ha registrato un aumento significativo. Le persone hanno cercato nuovi modi per intrattenersi a casa, e lo streaming musicale è diventato una delle attività principali.

Secondo i dati, nel 2021, la media settimanale di ascolto di musica era di 18,4 ore, rispetto alle 18 ore del 2019 e alle 17,8 ore del 2018 (**Gadget Advisor**). Questo incremento è dovuto in gran parte alla crescita delle piattaforme di streaming musicale come Apple Music e Spotify, che hanno visto un aumento significativo nel numero di utenti e nel tempo di ascolto. Ad esempio, nel primo trimestre del 2023, sono stati superati un trilione di stream audio in soli tre mesi (**Gadget Advisor**).

Le statistiche mostrano che quasi il 40% degli utenti di età compresa tra 35 e 64 anni ha utilizzato servizi di streaming musicale nell'ultimo mese, con una crescita particolarmente forte tra le generazioni più giovani, come i Millennial e la Generazione Z (**Comparitech**).

Queste fasce d'età sono le più propense a sottoscrivere abbonamenti a servizi di streaming musicale per godere di un'esperienza senza pubblicità, la possibilità di scegliere la musica da ascoltare e l'accesso a vaste librerie di brani (**Comparitech**).

L'industria della musica in streaming ha visto una crescita continua negli ultimi anni.

Nel 2022, il numero di brani ascoltati in streaming negli Stati Uniti ha raggiunto 1,3 trilioni, un aumento del 12,2% rispetto all'anno precedente (**Comparitech**). Inoltre, il mercato globale dello streaming musicale è cresciuto del 10,3% nello stesso anno, segnando l'ottavo anno consecutivo di crescita per l'industria musicale (**Comparitech**).

Questi dati evidenziano come la pandemia abbia accelerato l'adozione e l'uso delle piattaforme di streaming musicale, che continuano a crescere grazie alla crescente domanda di contenuti musicali on-demand da parte di un pubblico sempre più ampio e diversificato.

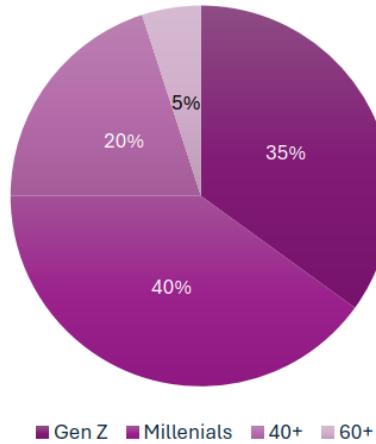
2.6.1 Definizione delle Personas

Lo studio delle **"personas"**, ovvero dei profili ideali di utenti, è fondamentale per comprendere gli elementi chiave dell'applicazione e orientare al meglio le sue funzionalità per soddisfare i desideri degli utenti.

Nel nostro caso specifico, il focus per individuare le user-personas è stata la fascia d'età, individuando quattro categorie principali:

- **Gen Z:** età compresa tra i 10-23 anni
- **Millenials:** età compresa tra i 24-39 anni
- **oltre i 40 anni**
- **oltre i 60 anni**

Distribuzione in percentuale degli utenti di una piattaforma di streaming musicale



Di seguito, verranno illustrate le *personas* ideate sulla base di quanto descritto.

- **Personas 1:** La prima user-persona appartiene alla fascia d'età denominata come Gen-Z. Studenti amanti della musica e della tecnologia che rappresentano la nuova generazione di utenti online.
- **Personas 2:** La seconda user-persona appartiene alla fascia d'età denominata come Millennials. Persone più affini alla tecnologia che sono appassionati di musica e che sono riusciti a farne di esse un mestiere o un'allegra compagnia durante le giornate.
- **Personas 3:** L'ultima user-persona appartiene al range d'età più estremo che abbiamo individuato. Questa classe prevede persone che non sono esperte di tecnologia ma che vivono della loro passione.



Hanah Vittoria

16 anni, Studentessa

“Non c’è nulla che tu possa fare che non sia possibile fare. Non c’è nulla che tu possa cantare che non sia possibile cantare.”

BIO

Nata a Milano da padre giapponese e madre italiana. Frequenta il terzo anno di liceo classico ma il suo sogno è quello di diventare una cantante famosa in tutto il mondo. Le piace stare in compagnia delle sue amiche dalle quali non si separa mai. Le piacciono gli animali, scrivere e comporre melodie.

LIKES

Ascoltare musica mentre studia

Il gelato a fragola

Suonare la chitarra

DISLIKES

La matematica

Fare attività fisica

Gli insetti

OBIETTIVI

Diventare una cantante famosa

Migliorare il suo inglese

Imparare a suonare il pianoforte

GENERI MUSICALI PREFERITI

K-pop

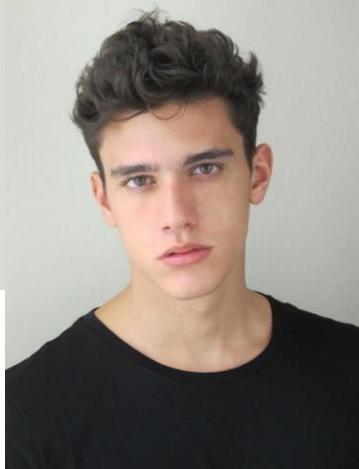
Pop

Indie Italiano

CANZONE PREFERITA

“Zen” by Pinguini Tattici

Nucleari



Sperti Cesare

30 anni, Personal-trainer

“Non si può battere la persona che non molla mai.”

BIO

Romano di nascita, ha una grande passione per lo sport e la musica. Fin da piccolo è stato molto atletico, partecipando a diverse attività sportive come calcio, nuoto e atletica leggera. Oggi lavora in palestra e corre per mantenersi in forma. Oltre allo sport, coltiva anche un grande amore per la musica; infatti nessuno dei suoi allenamenti inizia se non indossa le cuffie e fa partire la sua playlist preferita.

LIKES

Allenarsi all'aria aperta
Il sushi
Prendersi cura di sé

DISLIKES

I cavalli
Passare troppo tempo in casa
Giocare ai videogames

OBIETTIVI

Aprire la propria palestra
Visitare New York
Andare ad un concerto rock

GENERI MUSICALI PREFERITI

Rock
Metal
Rap
Hip-hop

CANZONE PREFERITA

“All Messed Up” by Sum 41



Galgani Gemma

74 anni, Ex-direttrice di teatro

"Il teatro è la più alta espressione dell'arte umana."

BIO

Ha dedicato oltre cinque decenni della sua vita al mondo delle arti sceniche. Ha iniziato la sua carriera come attrice prima di passare alla regia e infine alla direzione teatrale. Gemma è nota per il suo impegno nel promuovere giovani talenti e per la sua passione per il teatro sperimentale. Ora in pensione, continua ad essere una figura rispettata e influente nel panorama teatrale italiano.

LIKES

Opere drammatiche
Passeggiare
I cani

DISLIKES

Il caldo
Uscire di sera
Chi non ama i musical

OBIETTIVI

Diffondere l'amore per l'arte ai più giovani
Scrivere un'opera teatrale
Coltivare il proprio orto

GENERI MUSICALI PREFERITI

Opera classica
Cantautorato italiano

CANZONE PREFERITA

"La canzone dell'amor perduto" by Fabrizio De André

2.7 Valutazione dell’usabilità a priori

Per ottenere una valutazione dell’usabilità a priori più corretta possibile, si è cercato di riprodurre dei prototipi avendo le idee ben chiare sull’aspetto e sulle funzionalità che doveva avere l’applicativo. Si è deciso di restare il più fedeli possibile ai prototipi così da avere una valutazione a priori quanto più vicina a quella a posteriori in beta-testing. I mockup sono stati realizzati su **Figma** che consente di realizzare rapidamente prototipi interattivi, permettendo di testare e iterare facilmente le idee con gli stakeholder. È più semplice progettare interfacce accessibili e responsive utilizzando i potenti strumenti di layout e tipografia di Figma.

I mockup, e soprattutto l’app, sono stati realizzati seguendo le **8 regole d’oro di Shneiderman**.

NOTA: I punti evidenziati sono quelli sui quali è stata posta più attenzione.

- **Coerenza a tutti i costi**
- **Usabilità universale**
- Offrire riscontri informativi
- Dialogo con gli utilizzatori
- **Prevenire gli errori**
- Assicurare la reversibilità
- **Garantire il controllo degli utenti**
- **Ridurre il carico di memoria a breve termine**

2.7.1 Tabelle di valutazione e tecniche utilizzate

Precedentemente sono state chiarite le motivazioni dietro le scelte fatte in termini di design di UI e UX per l’applicativo. Pur ritenendo queste le migliori scelte, è altrettanto importante ottenere un feedback anche dai futuri utenti. Si è scelto quindi di fare una valutazione a priori dell’usabilità dell’applicativo, rispettando le **regole di Nielsen**². Sono stati selezionati **cinque candidati** ai quali è stato richiesto di eseguire **cinque task**, con pochissime indicazioni da parte del team.

- Registrazione di un account
- Creazione di una playlist
- Aggiunta di una canzone ad una playlist
- Modifica password
- Ricerca di un brano musicale

La scelta dei candidati è stata invece fatta sulla base della loro età e delle loro competenze digitali. Questi ultimi due valori saranno classificati in tale modo:

- **Naive:** useremo **N** per indicare ciò

²<https://aelaschool.com/en/interactiondesign/10-usability-heuristics-ui-design/>

- **Abile:** useremo **A** per indicare ciò
- **Esperto:** useremo **E** per indicare ciò

L'eventuale riuscita o meno delle task sarà segnata invece come:

- **F:** per indicare che la task non è stata completata
- **S:** per indicare che la task è stata completata con successo
- **//:** per indicare che la task è stata portata a termine, ma il candidato ha avuto bisogno di qualche aiuto in più su come procedere

Nome	Età	Sesso	Occupazione	Competenze digitali
<i>Marianna</i>	13	F	Studentessa scuole medie	N
<i>Luciana</i>	20	F	Studentessa universitaria	A
<i>Antonio</i>	26	M	Cybersecurity Analyst Junior	E
<i>Elisa</i>	35	F	Diretrice commerciale	E
<i>Giovanni</i>	50	M	Farmacista	A
<i>Giorgio</i>	68	M	Pensionato	N

Nome	Task 1	Task 2	Task 3	Task 4	Task 5
<i>Marianna</i>	S	S	S	//	S
<i>Luciana</i>	S	S	S	S	S
<i>Antonio</i>	S	S	S	S	S
<i>Elisa</i>	S	S	S	S	S
<i>Giovanni</i>	S	F	//	//	//
<i>Giorgio</i>	S	F	//	F	//

In generale si è piuttosto soddisfatti dei risultati. I candidati hanno tutti concordato sulla generale intuitività dell’interfaccia grafica e il numero di fallimenti è stato piuttosto limitato. Molto importante per il team è stato soprattutto notare come i due candidati adulti siano riusciti comunque a stabilire più successi che fallimenti, seppur necessitando spesso di conferme su quello che stavano facendo.

2.8 Prototipazione funzionale via statechart dell'interfaccia grafica

Di seguito vengono rappresentate mediante statechart **l'aggiunta di una playlist alle preferite** piattaforma e **l'eliminazione di una playlist**

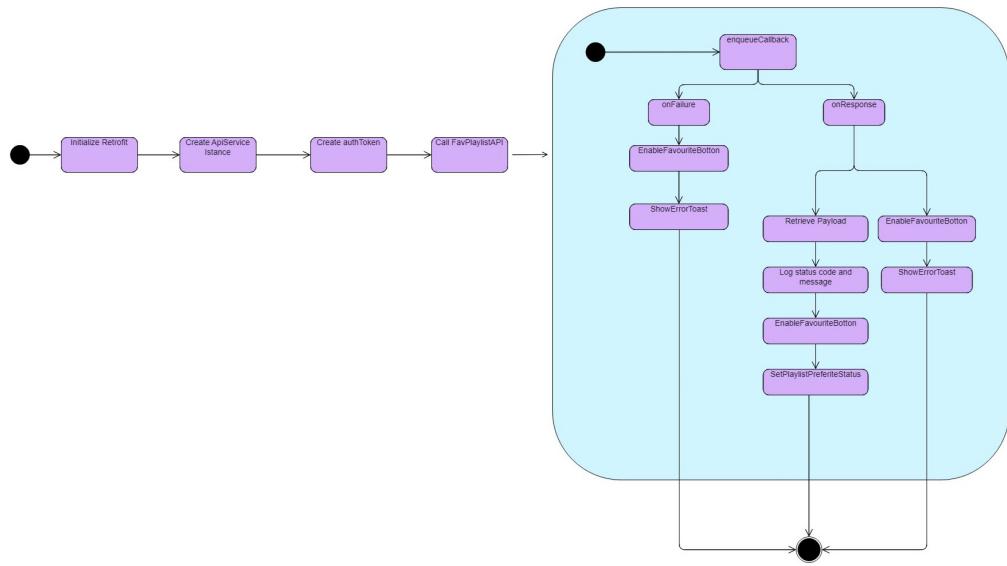


Figura 8: Statechart Inserisci playlist ai preferiti

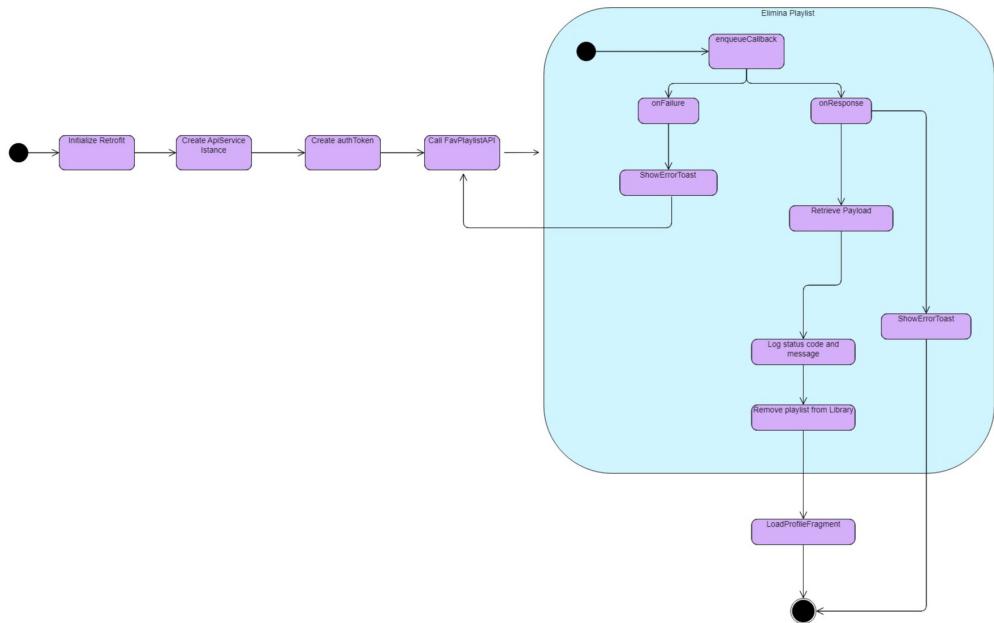


Figura 9: Statechart Elimina playlist

2.9 Glossario

2.9.1 Termini

- **Actor** Nelle tabelle di Cockburn o negli Use Case Diagram, è la persona che interagisce con il sistema
- **Database** Un archivio di dati strutturato per la memorizzazione persistente, la gestione e l'aggiornamento di informazioni utili per il sistema
- **Back-end** Parte di sviluppo che non opera sul client e che fornisce un servizio.
- **Android** Sistema operativo mobile
- **Mockup** È una rappresentazione grafica a scopo illustrativo di un oggetto o un sistema. Nel nostro caso, l'uso dei mock-up è funzionale a mostrare il funzionamento di specifiche interazioni col sistema
- **Tabelle di Cockburn** Descrizione in linguaggio naturale delle azioni necessarie all'esecuzione di un caso d'uso
- **Use Case** La rappresentazione di un'astrazione che descrive una classe di scenario del sistema
- **Statechart** Un diagramma per descrivere il comportamento di entità o di classi in termini di stato (macchina a stati)

- **UX Design** Tutto ciò che afferisce l'interazione tra una persona con un prodotto, un servizio o un sistema
- **Classe** Nella programmazione orientata a oggetti, una classe fa riferimento a un insieme di oggetti con proprietà comuni.
- **Framework** Architettura logica di supporto sulla quale un software può essere progettato e realizzato, facilitandone lo sviluppo.
- **Scalabilità** Capacità di un sistema di aumentare o diminuire di scala in funzione delle necessità.
- **Usabilità** Il "grado in cui un prodotto può essere usato da particolari utenti per raggiungere certi obiettivi con efficacia, efficienza e soddisfazione in uno specifico contesto d'uso.
- **Elasticità** Si riferisce alla capacità di un servizio cloud di offrire servizi su richiesta aumentando o diminuendo le risorse quando la domanda sale o scende.

3 Modelli di dominio

In questa sezione verranno analizzati i modelli di dominio dell'applicazione ancora in fase di analisi dei requisiti. In particolare utilizzeremo per tale analisi:

- **Class Diagram**
- **Sequence Diagram**
- **Diagramma di attività**

Per far sì che ci sia una facile comprensione e per individuare classi ed entità in gioco è stata ritenuta fondamentale l'euristica **Three Object-Type** o anche conosciuta con tre acronimi:

- **Entity:** entità o modelli che sono in gioco
- **Boundary:** rappresentano le interazioni tra sistema ed utente
- **Control:** rappresentano i controller o meglio definita la logica del sistema che si occupa di definire gli use case sotto sviluppo

3.1 Classi, oggetti e relazioni di analisi

3.1.1 Classi ed entità

Di seguito verrà riportata una rappresentazione delle entità che sono state individuate e che saranno esse stesse gli oggetti in gioco per i vari use case assegnati.

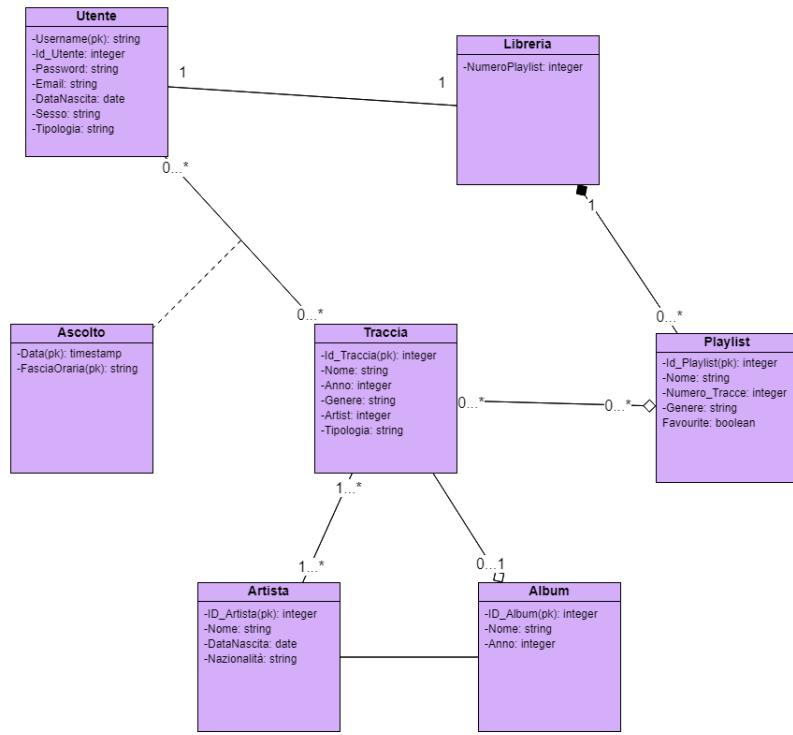


Figura 10: Class Diagram generale

3.1.2 Class diagram delle funzionalità

In questa sezione elencheremo tutti i classdiagram suddivisi.

NOTA: Per una questione puramente stilistica volta ad una comprensione facilitata dei class diagram, si è optato per utilizzare bianco e nero, piuttosto che il viola.

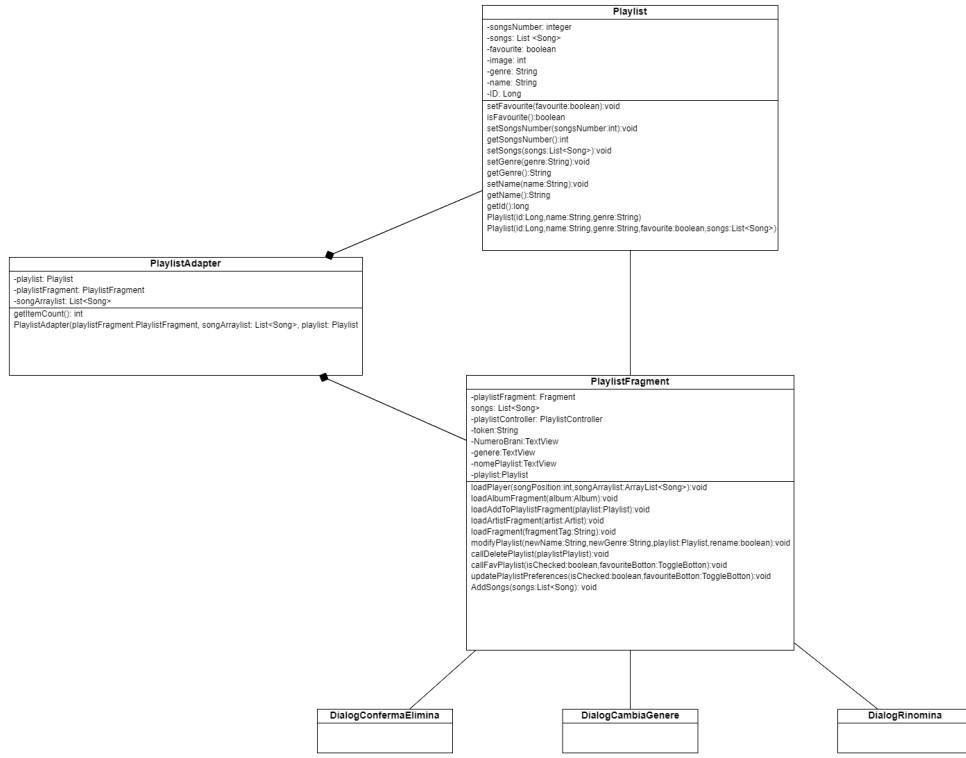


Figura 11: Class Diagram Playlist

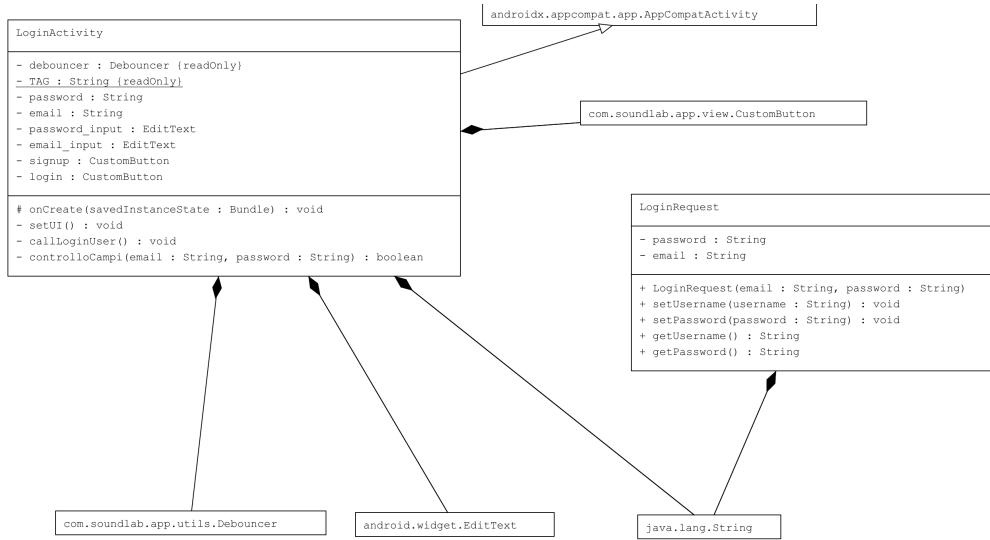


Figura 12: Class Diagram Login

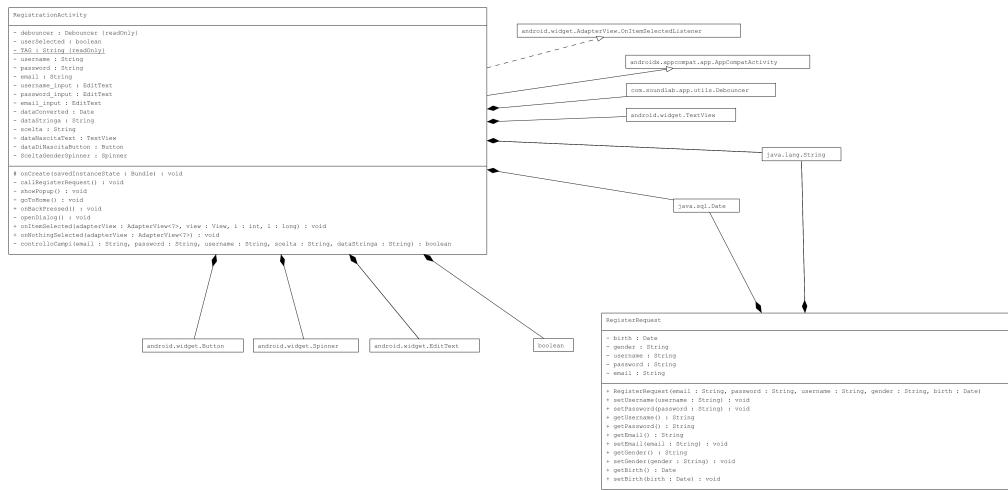


Figura 13: Class Diagram Registrazione

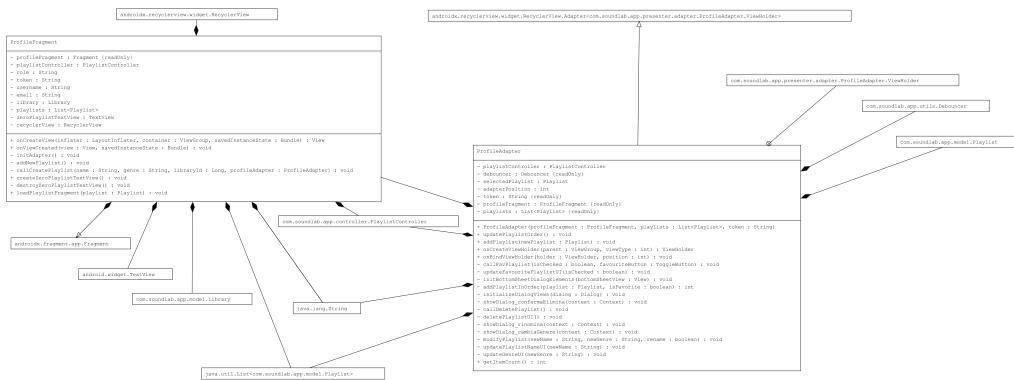


Figura 14: Class Diagram Profilo

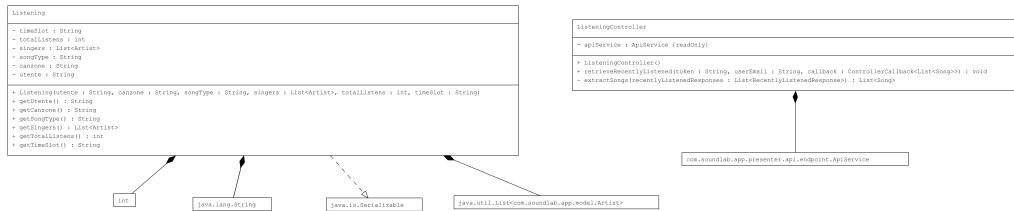


Figura 15: Class Diagram Listening

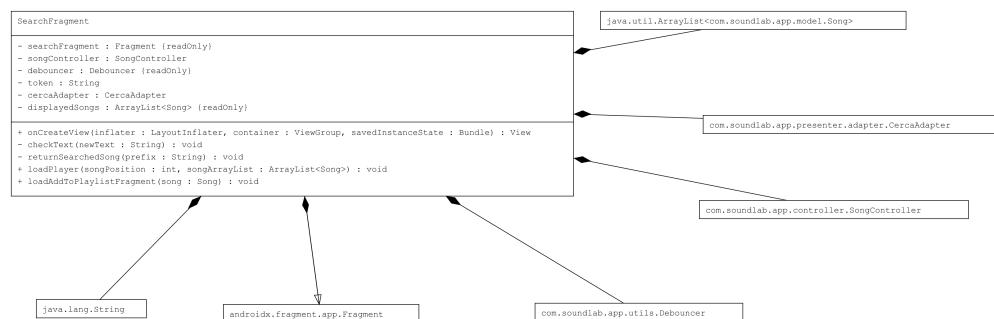


Figura 16: Class Diagram Search

3.2 Sequence diagram

Un **Sequence Diagram** è un diagramma previsto dall'UML utilizzato per descrivere uno **scenario**. Uno scenario è una determinata sequenza di azioni in cui tutte le scelte sono state già effettuate.

3.2.1 Sequence Diagram Aggiungi playlist

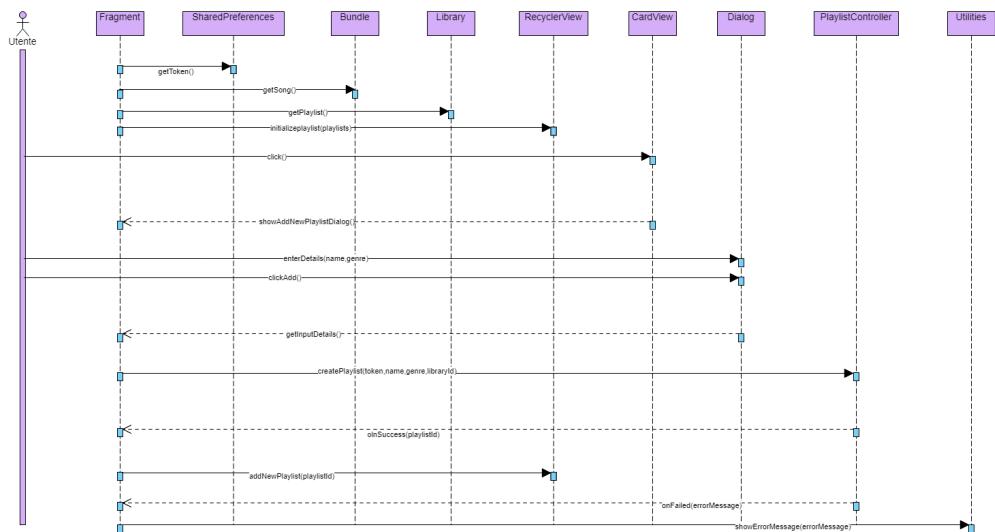


Figura 17: Class Diagram Aggiungi playlist

3.2.2 Sequence Diagram Elimina profilo

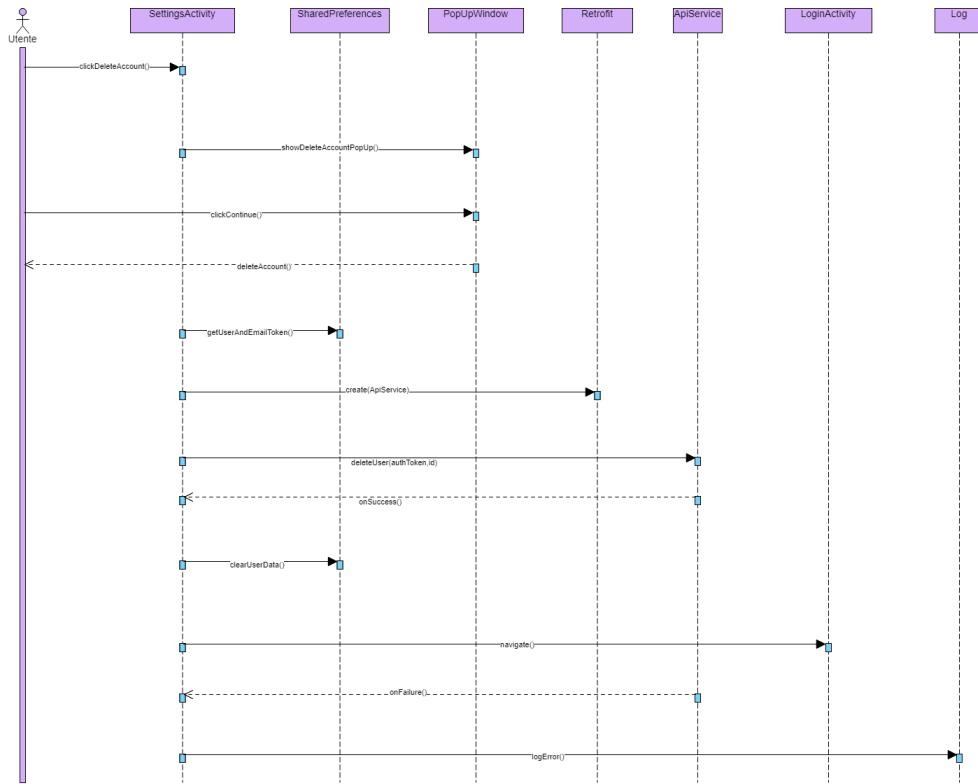


Figura 18: Class Diagram Elimina Profilo

3.3 Activity diagram

Il **diagramma di attività** è un tipo di diagramma che permette di descrivere un processo attraverso dei grafi in cui i nodi rappresentano le attività e gli archi l’ordine con cui vengono eseguite.

3.3.1 Activity diagram

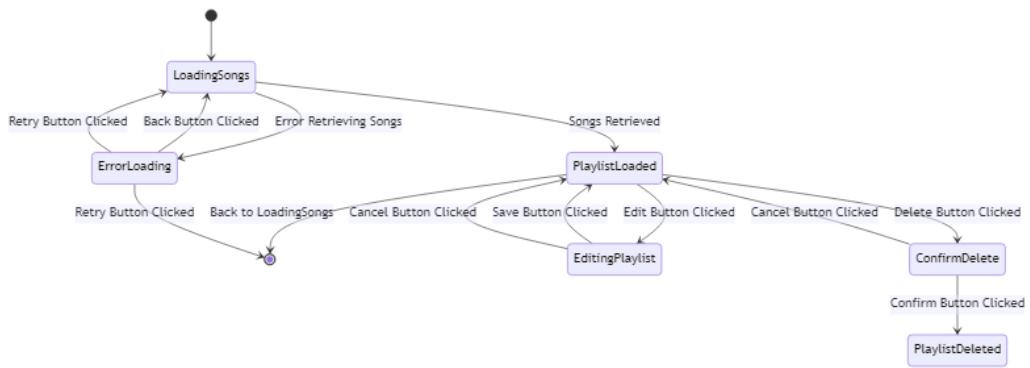


Figura 19: Activity Diagram funzioni playlist

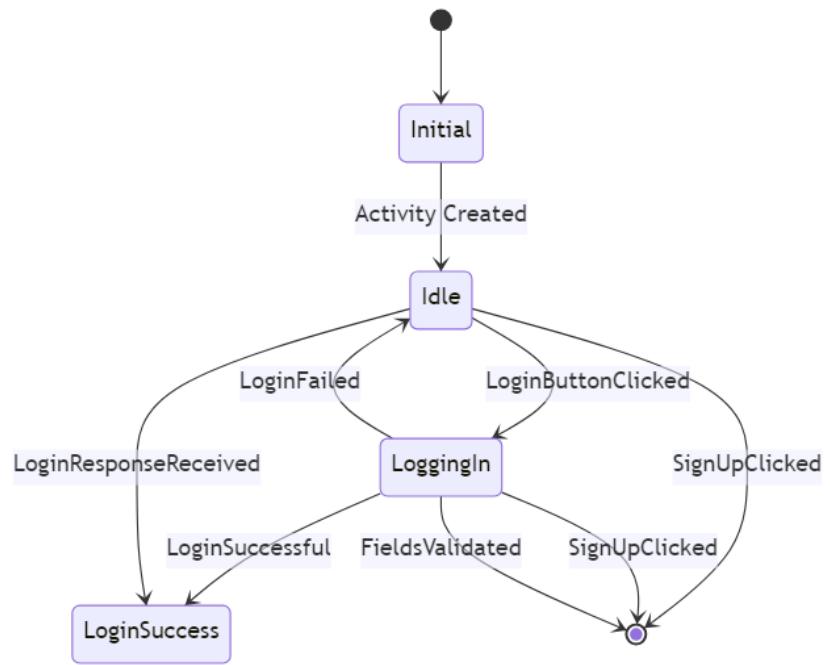


Figura 20: Activity Diagram Login

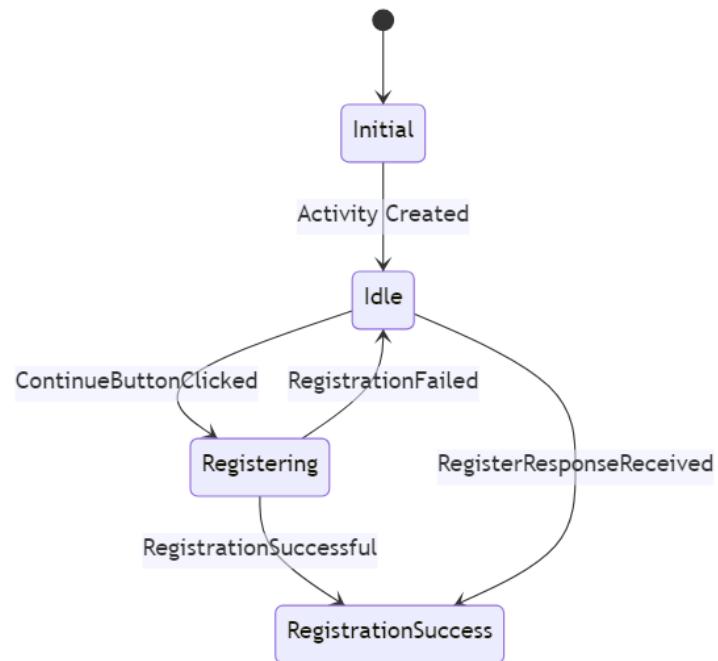


Figura 21: Activity Diagram Registrazione

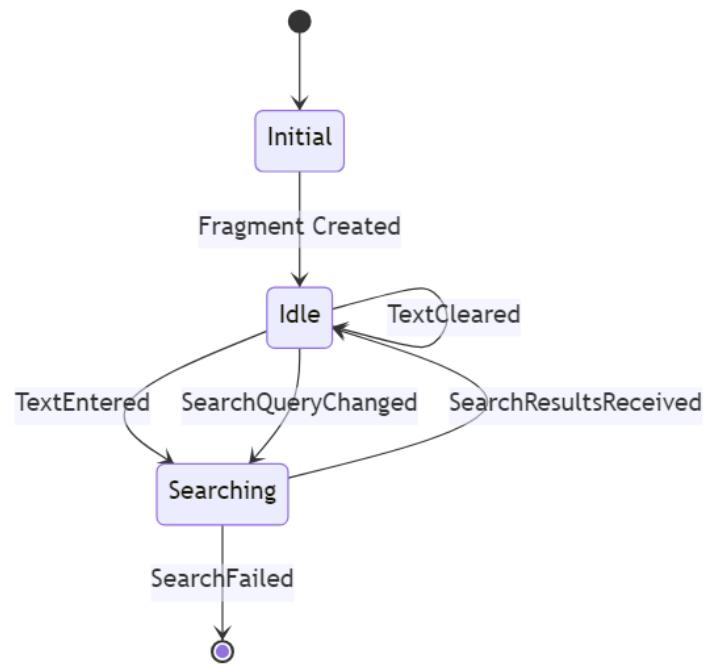


Figura 22: Activity Diagram Ricerca

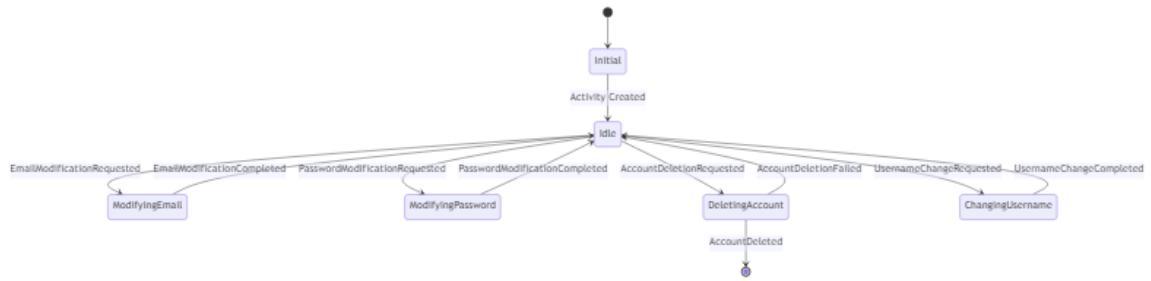


Figura 23: Activity Diagram azioni settings

4 Design di sistema

4.1 Analisi architetturale

L'architettura usata per questo progetto comprende un insieme di soluzioni e strategie di famosi **design pattern**.

In particolare, in questa sezione del documento, verrà analizzata l'**architettura client-server** utilizzata.

Con la parola *client*, indichiamo i dispositivi android mobile mentre con la parola *server*, indichiamo il backend utilizzato per gestire i servizi offerti ai clienti dal nostro applicativo.

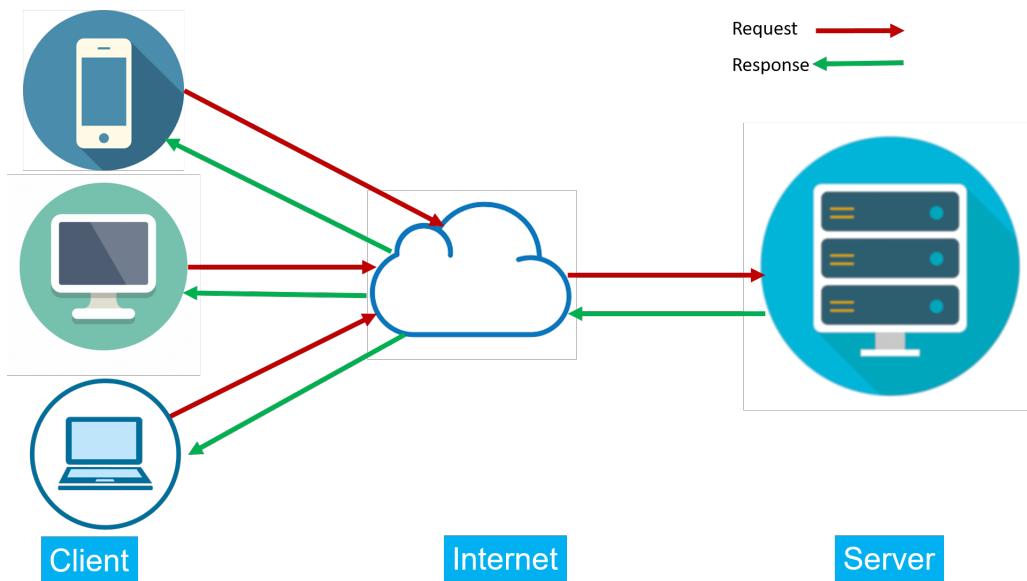


Figura 24: Client-server model

4.1.1 Descrizione architettura cloud

SoundLab offre un back-end con le seguenti caratteristiche:

- *Elasticità*
- *Scalabilità*
- *Economicità*
- *Astrazione*
- *Sicurezza*
- *Virtualizzazione*

Tali servizi sono stati offerti grazie all'utilizzo dei **servizi on demand in Cloud di AWS**³, applicando la seguente architettura:

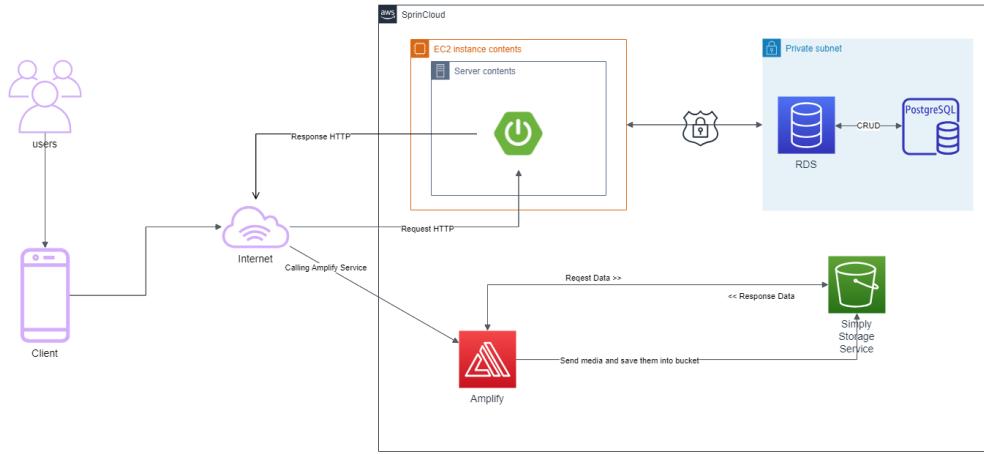


Figura 25: Architettura di sistema

Le tecnologie adoperate sono:

- **Springboot** che risulta essere un tassello importantissimo per il backend. Questo famosissimo framework di Java offre l'opportunità di creare un proprio insieme personalizzato di **route** di **REST API** personalizzate. Inoltre, SpringBoot è il diretto connesso al server Database tramite operazioni **CRUD**.
- **Amazon AWS-RDS** che è un sistema che semplifica l'impostazione, il funzionamento e il dimensionamento di database relazionali nel cloud. Questo servizio fornisce una capacità ridimensionabile efficiente nei costi, automatizzando al tempo stesso le attività di amministrazione più dispendiose in termini di tempo.
- **Amazon AWS-S3** che è un servizio di archiviazione di oggetti che offre scalabilità, disponibilità dei dati, sicurezza e prestazioni all'avanguardia nel settore. I clienti di tutte le entità e settori possono archiviare e proteggere qualsiasi quantità di dati per qualsiasi caso d'uso. Con classi di archiviazione convenienti e caratteristiche di gestione di facile utilizzo, è possibile ottimizzare i costi, organizzare i dati e configurare controlli di accesso ottimizzati per soddisfare specifici requisiti aziendali, organizzativi e di conformità.
- **Amazon AWS-Amplify** che consiste in un set di strumenti e caratteristiche appositamente progettati per consentire agli sviluppatori front-end di applicazioni Web e per dispositivi mobili di costruire rapidamente e facilmente applicazioni full-stack in AWS, con la flessibilità di sfruttare i vari servizi AWS man mano che i casi d'uso si evolvono. Con Amplify, è possibile configurare un back-end per app Web operanti su dispositivi mobili, connettere l'app in

³Amazon Web Services, Inc. è una sussidiaria di Amazon, che fornisce servizi di cloud computing su un'omonima piattaforma on demand

pochi minuti, costruire visivamente un'interfaccia utente front-end Web e gestire facilmente i contenuti dell'app al di fuori della console AWS

4.1.2 Il server

Il server è scritto interamente in **Springboot**. Si tratta di un framework open source di livello aziendale molto diffuso per la creazione di applicazioni autonome, adatte ad ambienti di produzione che vengono eseguite su **JVM (Java Virtual Machine)**.

Per una buona gestione e per facilitare tante qualità quali il riutilizzo di codice, dependency injection e altro abbiamo optato per un famoso **DesignPattern**, ovvero **MVC**.

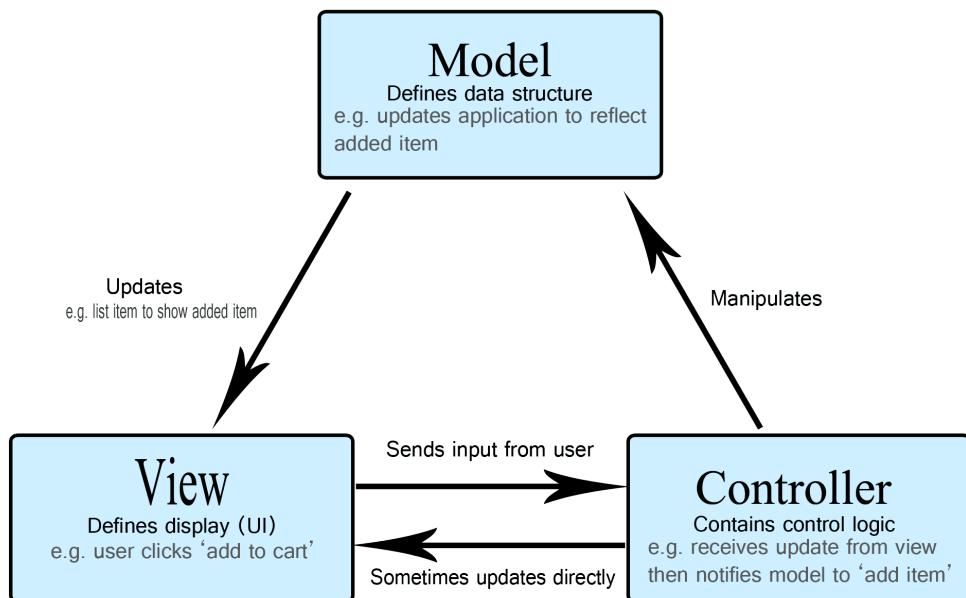


Figura 26: Design Pattern MVC

Il design pattern segue delle linee guida per SpringBoot, infatti sono stati individuati i seguenti componenti:

- **Components:** componenti del backend quali controller, service, repository (tutte sottoclassi)
- **Model:** oggetti che rappresentano una figura concreta che mappata viene per essere entità di un database
- **Model DTO:** *Data Transfer Object* sono il mezzo di comunicazione entrante/uscente del server

- **Controller:** gestiscono i metodi da HTTP (GET, POST, PUT, DELETE) e relative rotte e sotto-rotte
- **Service:** layer soggetto a dependency injection molto importante e intermediario tra repository e controller
- **Repository:** diretta comunicazione con la logica di database sottostante, attraverso JPA e altri moduli si interfacciano con interfacce per fare operazioni CRUD

Nel prossimo paragrafo elencheremo le **REST API** create appositamente per offrire i servizi tramite **Springboot** per i clienti di SoundLab.

4.1.3 Rest Api

Il protocollo di comunicazione è quello **HTTP**.

- **POST:** Il metodo POST viene utilizzato per inviare dati al server per creare o aggiornare risorse. Quando si effettua una richiesta POST, i dati vengono solitamente inviati nel corpo della richiesta. Questo metodo è comunemente utilizzato per inviare i dati di un modulo HTML ad un server o per creare nuove risorse tramite un'API.
- **GET:** Il metodo GET viene utilizzato per richiedere dati dal server. Quando si effettua una richiesta GET, i parametri e i dati della richiesta sono inclusi nell'URL. Questo metodo è comunemente utilizzato per ottenere risorse esistenti dal server o per recuperare dati da un'API.
- **DELETE:** Il metodo DELETE viene utilizzato per eliminare una risorsa specifica dal server. Quando si effettua una richiesta DELETE, l'identificatore della risorsa da eliminare è solitamente incluso nell'URL o nel corpo della richiesta. Questo metodo è comunemente utilizzato per cancellare risorse esistenti tramite un'API.

Campo	Route	Descrizione
loginUser	/api/data/users/{id}	Effettua una richiesta POST per autenticare un utente
registerUser	/api/authentication/register	Effettua una richiesta POST per registrare un nuovo utente
userLib	/api/data/libs/{id}	Effettua una richiesta GET per ottenere la libreria dell'utente in base all'ID
createPlaylist	/api/data/playlist/createPl	Effettua una richiesta POST per creare una playlist
deletePlaylist	/api/data/playlist/{id}	Effettua una richiesta DELETE per eliminare una playlist in base all'ID
modifyPlaylist	/api/data/playlist/renamePl	Effettua una richiesta POST per modificare una playlist
favPlaylist	/api/data/playlist/toggleFav/{id}	Effettua una richiesta POST per aggiungere o rimuovere una playlist dai preferiti.
deleteUser	/api/data/users/{id}	Effettua una richiesta DELETE per eliminare un utente
recentlyListened	/api/data/listenings/recently/{id}	Effettua una richiesta GET per ottenere la lista di brani ascoltati di recente dell'utente
searchSong	/api/data/song/search/{prefix}	Effettua una richiesta GET per cercare brani in base a un prefisso
changePw	/api/authentication/changepw	Effettua una richiesta POST per cambiare la password di un utente
insertSong	/api/data/playlist/addToPl	Effettua una richiesta POST per aggiungere una canzone a una playlist
deleteSong	/api/data/playlist/delFrPl	Effettua una richiesta DELETE per rimuovere una canzone da una playlist

NOTA: oltre a **changePw**, sono state implementate come REST API anche “**changeusrn**” e “**changemail**”, rispettivamente utilizzate per modificare il nome utente e per cambiare l’e-mail dell’utente.

4.1.4 Il client

Il client di SoundLab è stato sviluppato su piattaforma **Android**. I vantaggi di lavorare su android sono molteplici uno tra quali l'**Open Source**, infatti grazie ad android si ha avuto la possibilità di avere accesso a numerose librerie utili al fine di sviluppare al meglio l'applicativo. Per il progetto abbiamo usato il design pattern **MVP**:

- **Model:** rappresenta il modello dei dati di interesse per l'applicazione. Tale livello si occupa di incapsulare lo stato dell'applicazione, gestisce l'accesso alla sorgente dei dati, fornisce funzionalità per l'aggiornamento dello stato e l'accesso ai dati. Nel client android sviluppato, è compito di questo livello gestire la comunicazione diretta con le APILambda e quindi si occupa della formattazione delle richieste e della decodifica delle risposte. Infine, il MODEL notifica al PRESENTER il cambiamento di stato
- **View:** questo livello rappresenta l'interfaccia utente, permettendo l'interazione con esso e fornendo una rappresentazione grafica ed interattiva del model. La responsabilità di questo livello è la presentazione dei dati e dello stato dell'applicazione. Inoltre, esso riceve notifiche dal PRESENTER e aggiorna la visualizzazione
- **Presenter:** tale livello definisce la logica di controllo e le funzionalità applicative. Quindi, è compito di questo livello gestire gli eventi ed i comandi generati dall'utente: in base a questi ultimi, esso opera sul model. Infine, tale livello si occupa di selezionare/aggiornare il livello VIEW in base ai dati recuperati.

MVP

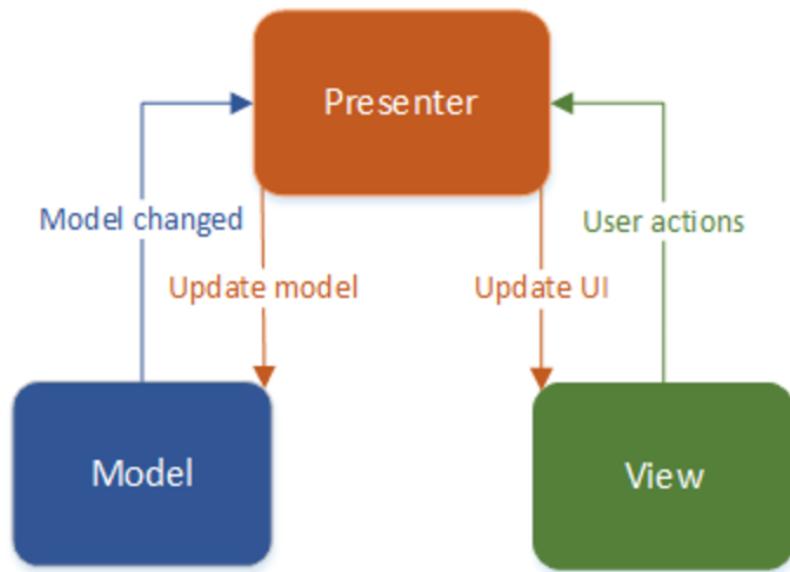


Figura 27: Design Pattern MVP

4.1.5 Supporti

Retrofit è una libreria per Android che semplifica l'utilizzo delle API RESTful all'interno di un progetto. Fornisce un modo semplice e intuitivo per definire le richieste HTTP, gestire le risposte del server e convertire automaticamente i dati JSON in oggetti Java.

- **Interfacce API:** In Retrofit, si definiscono le API RESTful come interfacce Java. Ogni metodo dell'interfaccia rappresenta un'operazione su una risorsa del server e specifica il tipo di richiesta HTTP (GET, POST, DELETE, ecc.), l'URL relativo e i parametri richiesti. Si possono anche specificare gli header di autenticazione o il corpo della richiesta utilizzando annotazioni specifiche di Retrofit.
- **Client Retrofit:** Si può creare un'istanza di Retrofit utilizzando il Retrofit.Builder e configurarlo con l'URL di base per le API. Il client Retrofit si occupa di gestire le richieste e le risposte HTTP, nonché della conversione automatica dei dati JSON in oggetti Java utilizzando un convertitore (come Gson o Jackson).
- **Convertitori:** Retrofit supporta diversi convertitori per la conversione automatica dei dati tra il formato JSON e gli oggetti Java. Si può scegliere il convertitore preferito e configurarlo con Retrofit. Gson è il convertitore predefinito di Retrofit.
- **Chiamate asincrone:** Retrofit esegue richieste HTTP in modo asincrono per non bloccare il thread principale dell'applicazione.
- **Esecuzione delle richieste:** Per eseguire effettivamente una richiesta HTTP, si chiama il metodo corrispondente dell'interfaccia API tramite l'istanza Retrofit creata. Retrofit si occuperà di creare e inviare la richiesta HTTP al server, gestire la risposta e restituirla al tuo codice.
- **Gestione degli errori:** Retrofit offre un modo semplice per gestire gli errori nelle chiamate alle API. Si possono definire classi di risposta personalizzate che rappresentano le risposte previste dal server e gestire anche gli errori di rete o di parsing dei dati. Si possono utilizzare annotazioni come `@HTTP`, `@Headers` e `@Path` per gestire situazioni specifiche come richieste personalizzate o parametri dinamici nell'URL.

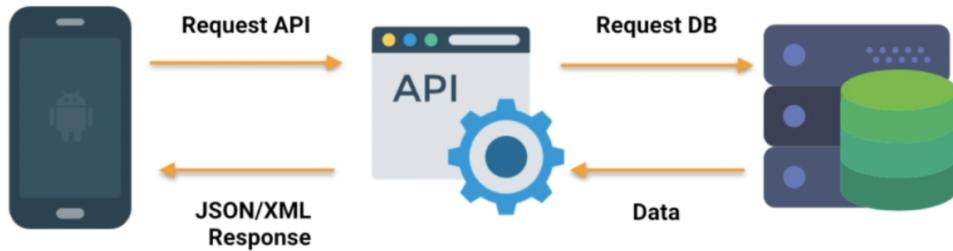


Figura 28: Android Retrofit

Firebase è una piattaforma di sviluppo di applicazioni mobile e web fornita da Google.

Offre una suite completa di strumenti e servizi progettati per aiutare gli sviluppatori a creare app di alta qualità, espanderne l'utenza e aumentare i ricavi.

La piattaforma è particolarmente apprezzata per la sua capacità di semplificare molte delle operazioni necessarie per costruire, gestire e scalare le applicazioni.

Le principali caratteristiche sono:

- **Realtime Database:** Un database NoSQL che consente di archiviare e sincronizzare i dati tra i client in tempo reale.
- Cloud Firestore: Un database flessibile, scalabile e di nuova generazione per lo sviluppo mobile, web e server.
- **Authentication:** Soluzioni di autenticazione per gli utenti tramite email, password e provider di terze parti come Google, Facebook e Twitter.
- **Hosting:** Hosting web statico veloce e sicuro per le tue app, con supporto per contenuti globalmente distribuiti tramite CDN.
- **Cloud Functions:** Funzionalità serverless che consentono di eseguire codice backend in risposta a eventi attivati da Firebase o richieste HTTP.
- **Cloud Messaging:** Un servizio che permette l'invio di notifiche push e messaggi agli utenti su diverse piattaforme.
- **Analytics:** Strumenti di analisi avanzati che forniscono una comprensione approfondita del comportamento degli utenti e delle prestazioni dell'app.
- **Crashlytics:** Un servizio di reporting in tempo reale per crash e bug, che aiuta a migliorare la stabilità dell'app.
- **Remote Config:** Permette di modificare dinamicamente l'aspetto e il comportamento dell'app senza rilasciare una nuova versione.

- **Performance Monitoring:** Monitoraggio delle prestazioni delle app in tempo reale, aiutando a identificare e risolvere i problemi di latenza e di consumo delle risorse.

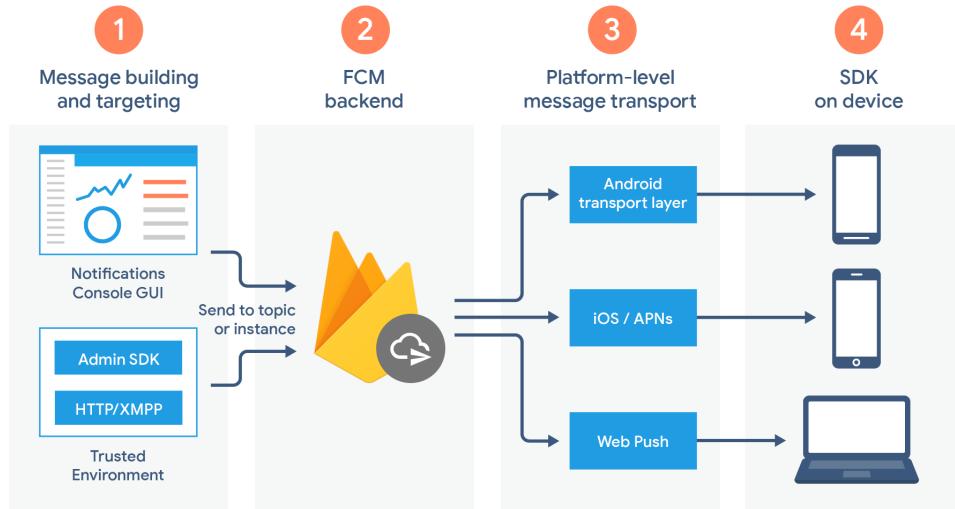


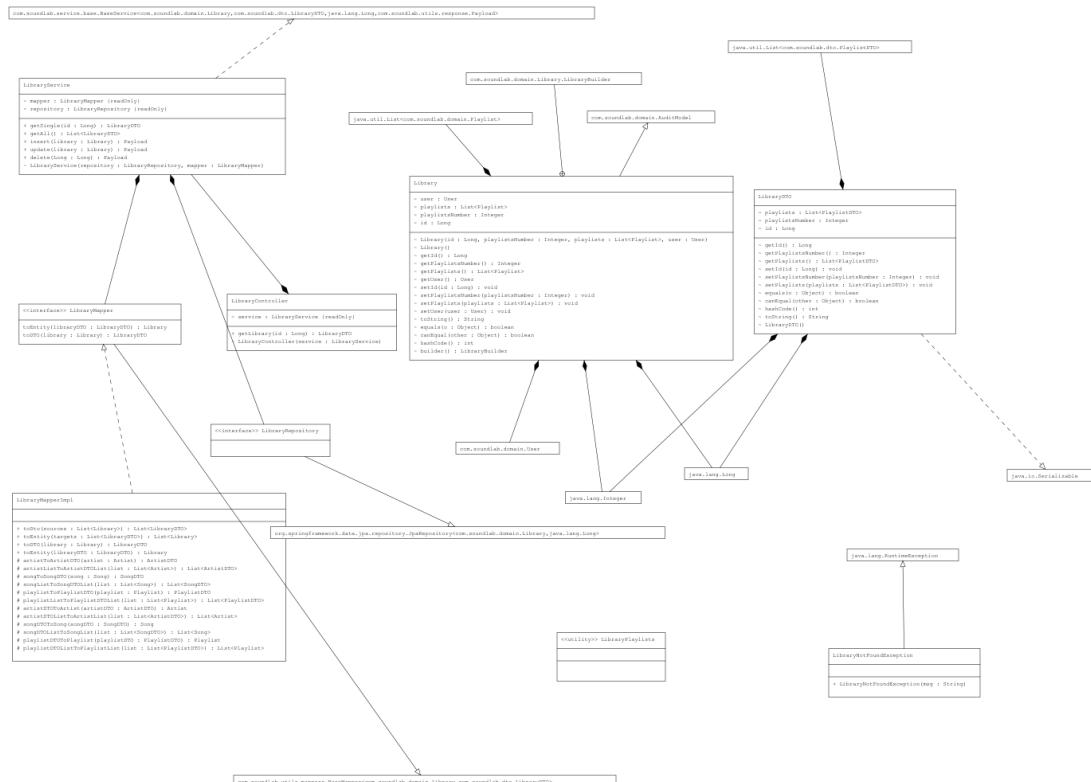
Figura 29: Firebase

4.2 Class Diagram di Design

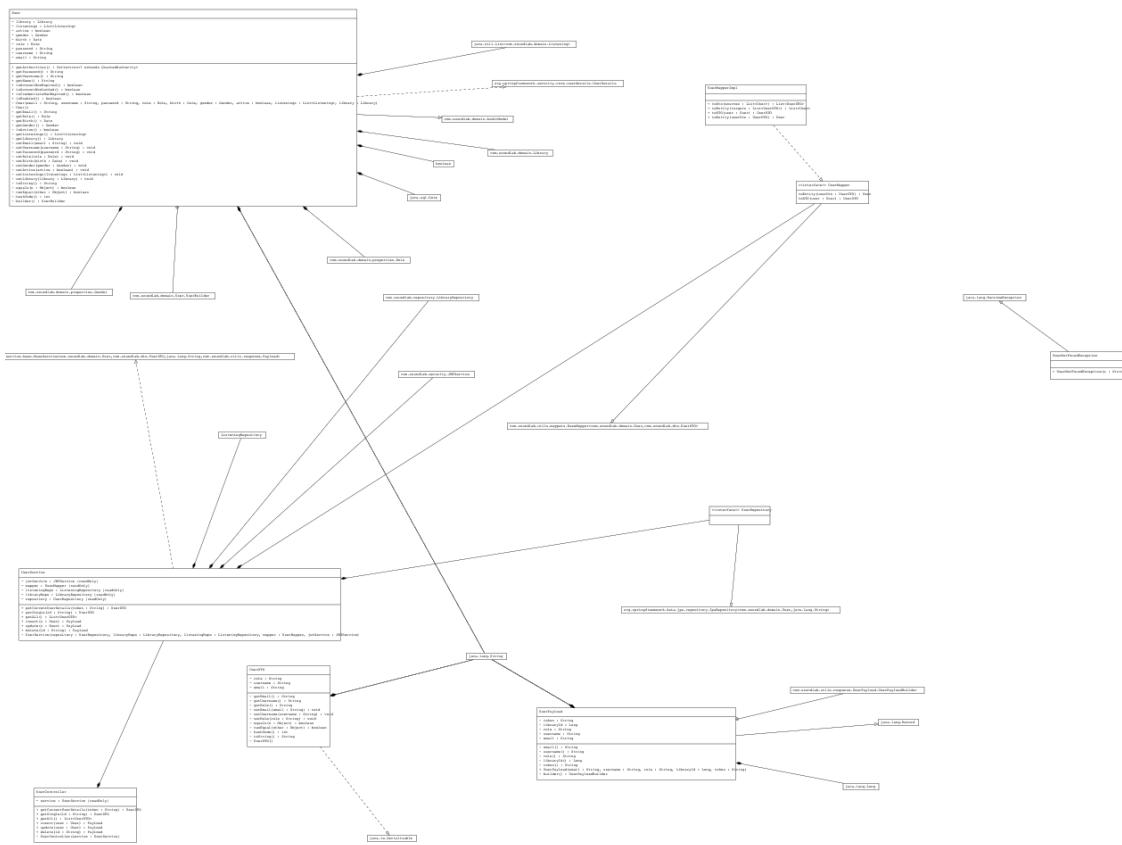
Per questioni di leggibilità e comprensione dei diagrammi, si è scelto di dividere il class diagram, creandone uno per ciascun package dell'applicativo.

NOTA: Trattandosi di diagrammi molto ricchi, si consiglia di zoomare per poter leggere bene il contenuto in quanto potrebbe risultare poco visibile.

4.2.1 Classi Diagram Playlist



4.2.2 Class Diagram User



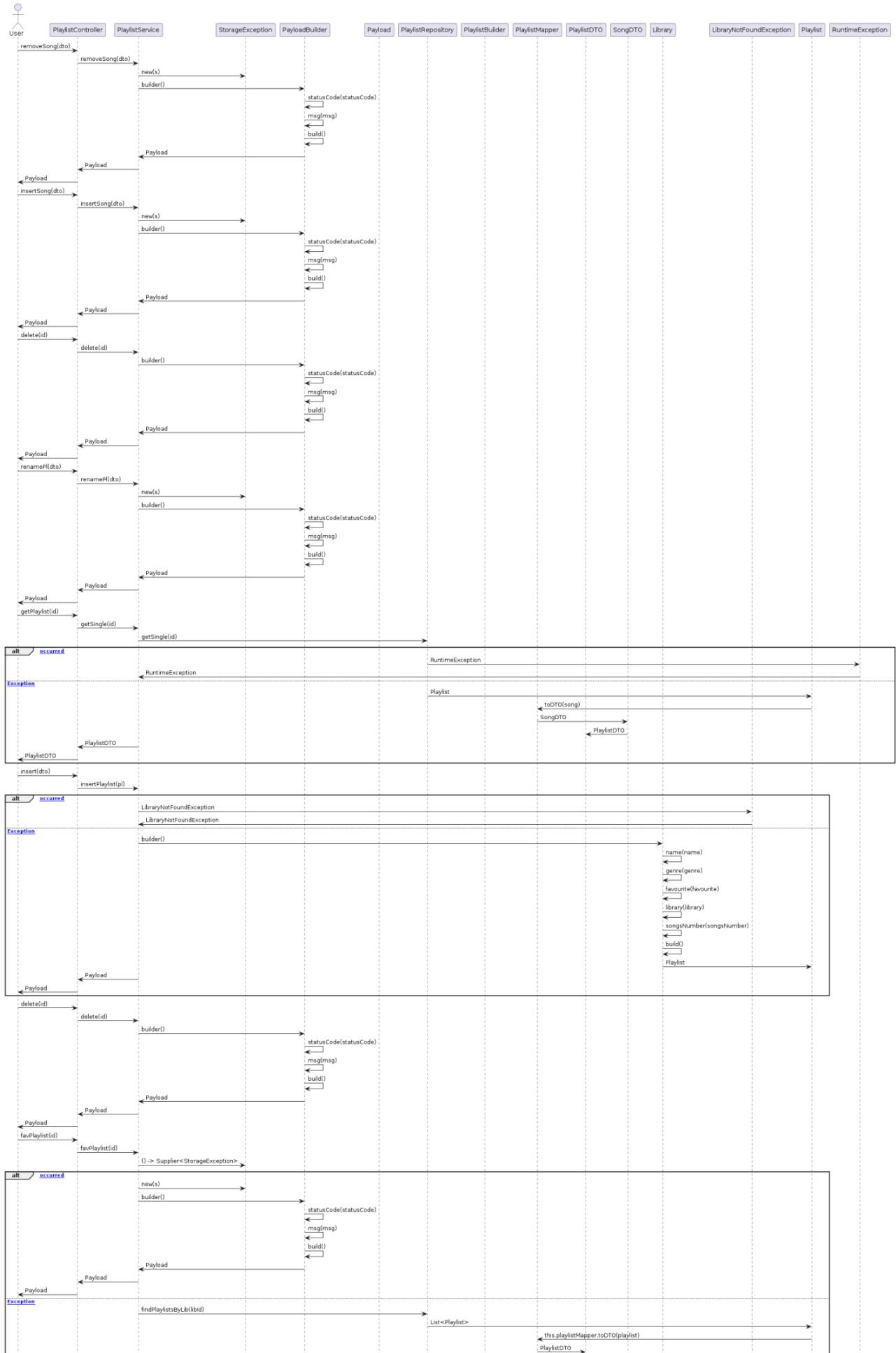
4.2.3 Class Diagram Library

4.2.4 Class Diagram Listening

4.3 Sequence Diagram di design

Per questioni di leggibilità e comprensione dei diagrammi, si è scelto di dividere i sequence diagram.
NOTA: Trattandosi di diagrammi molto ricchi, laddove sia necessario, si consiglia di zoomare per poter leggere bene il contenuto in quanto potrebbe risultare poco visibile.

4.3.1 Sequence diagram Playlist



4.3.2 Sequence diagram User

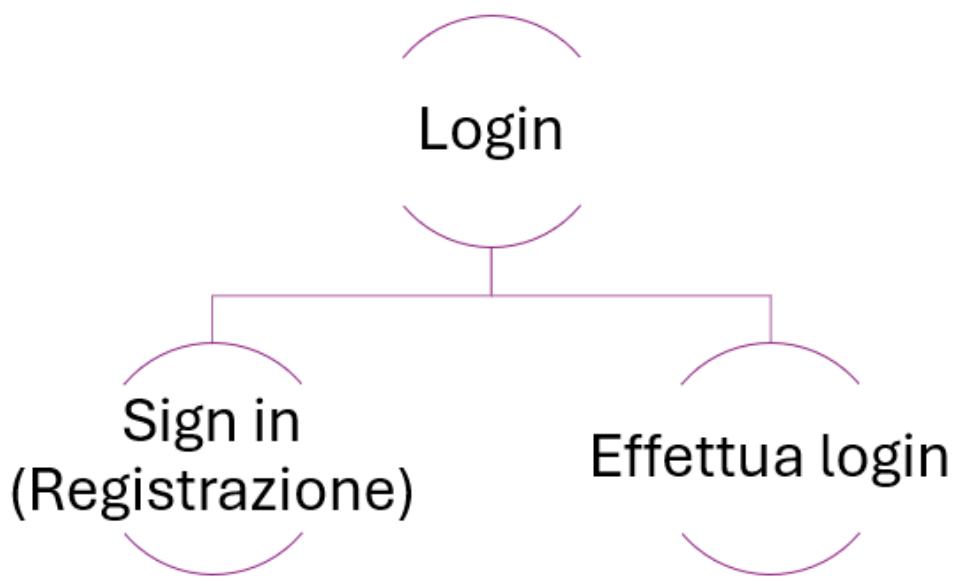


4.4 Gerarchie funzionali

La gerarchia funzionale è un metodo per organizzare e rappresentare le funzioni di un sistema in modo gerarchico. Questo approccio permette di visualizzare le relazioni tra le diverse funzioni e sottosistemi, evidenziando come si suddividono i compiti e le responsabilità.

In questo paragrafo vengono rappresentate le gerarchie funzionali dell'applicativo; per semplicità sono state suddivise in due macro categorie che sono state gestite come insiemi a parte ma comunicanti in qualche modo tra di loro.

4.4.1 Gerarchie funzionali Login



4.4.2 Gerarchie funzionali Homepage

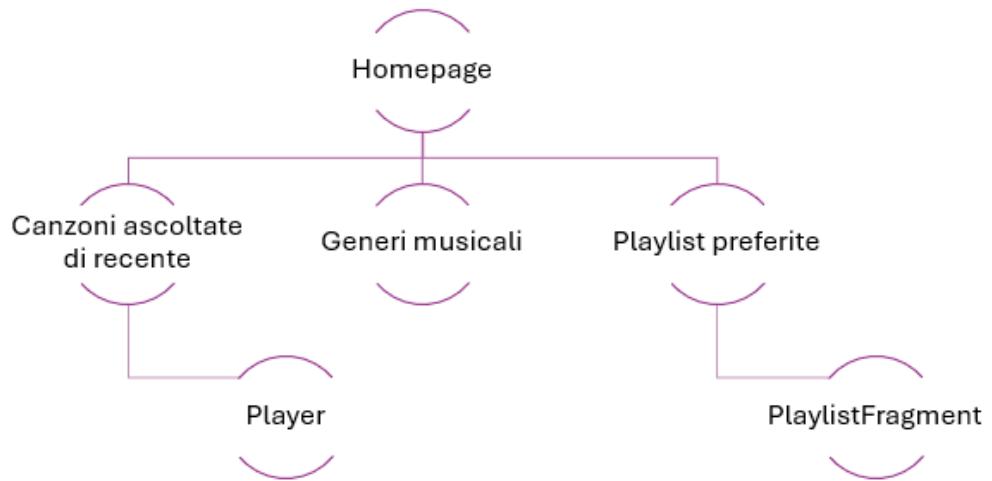


Figura 30: Gerarchia HomePage prima parte

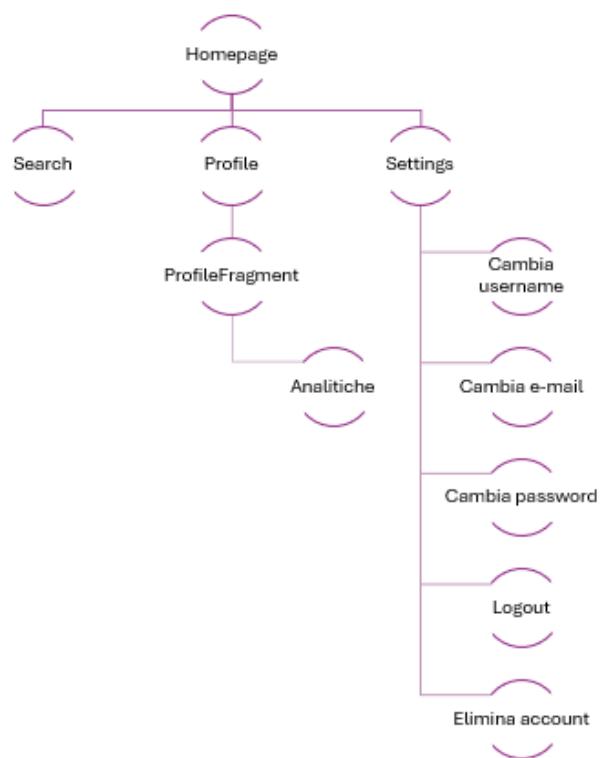


Figura 31: Gerarchia HomePage seconda parte

5 Codice xUnit

Nel contesto dello sviluppo software, l'importanza dei test unitari è fondamentale per garantire la qualità e la robustezza delle applicazioni.

Nel seguente capitolo, verranno esplorati due approcci principali per il testing del software: **Black Box Testing** e **White Box Testing**. Risulta essenziale comprendere il concetto di **test unitario** e il ruolo cruciale che svolgono nel processo di sviluppo.

Verrà preso in considerazione l'esempio del framework JUnit, un pilastro nel mondo del testing per il linguaggio di programmazione Java. Successivamente, verranno analizzate due strategie di testing sopra citate, evidenziando le differenze chiave tra di esse e illustrando l'implementazione pratica attraverso codici di test.

Si definisce, come primo passo, la differenza tra le due metodologie di testing:

- **Black box:**

- In questo approccio, il tester tratta il sistema come una "scatola nera", senza conoscere i dettagli interni del suo funzionamento. Il tester si concentra esclusivamente sui requisiti funzionali del sistema e sulle sue interfacce esterne.
- I test black box sono progettati per verificare se il sistema produce gli output desiderati dati certi input, senza preoccuparsi di come il sistema raggiunge tali output.
- Questo tipo di testing è utile per identificare bug, discrepanze tra requisiti e implementazione, e comportamenti imprevisti.

- **White box:**

- Questo approccio si concentra invece sulle strutture interne del software, osservando il codice sorgente e i dettagli della sua implementazione.
- I test white box vengono progettati per esplorare il flusso di controllo del programma, il flusso di dati, e per verificare la correttezza delle strutture di dati e algoritmi.
- È particolarmente utile per garantire una buona copertura del codice, identificare bug logici e ottimizzare le prestazioni del software.

5.1 Test InsertOneListening()

```
1  @Test
2  public void testInsertOneListening() {
3      AddRemoveListeningDTO dto = new AddRemoveListeningDTO(
4          mockUser.getEmail(), mockSong.getId());
5      Payload result = listeningService.insertListening(dto);
6
7      assertEquals(HttpStatus.OK.value(), result.statusCode());
8      assertEquals("Canzone ascoltata con successo", result.msg());
9
10     verify(userRepository).findById("mock.user@mail.com");
11     verify(songRepository).findById(1L);
12     verify(listeningRepository).save(any(Listening.class));
13 }
```

5.2 Test RemoveSongFromPlaylist()

5.3 Test InsertOneListening

```
1  @Test
2  public void testRemoveSongFromPlaylist(){
3      AddRemoveSongDTO dto = new AddRemoveSongDTO(mockPlay.getId()
4          (), mockSong.getId());
5      Payload result = playlistService.removeSong(dto);
6
7      assertEquals(HttpStatus.OK.value(), result.statusCode());
8      assertEquals("Canzone rimossa con successo", result.msg());
9      assertEquals(0, mockPlay.getSongsNumber());
10
11     verify(playlistRepository).findById(1L);
12     verify(songRepository).findById(1L);
13     verify(playlistRepository).save(any(Playlist.class));
14 }
```

5.4 Test testRegistration()

```
1  @Test
2  public void testRegistration(){
3
4      CredentialsDTO credentialsDTO = new CredentialsDTO("test@example.com", "password", "testUser", "Male", Date.
5          valueOf("1990-01-01"));
6
7      User user = User.builder()
8          .email(credentialsDTO.email())
9          .password("encodedPassword")
10         .username(credentialsDTO.username())
11         .role(Role.USER)
12         .birth(credentialsDTO.birth())
13         .gender(Gender.Male)
14         .active(true)
15         .build();
16
17      Library library = Library.builder()
18          .user(user)
19          .playlistsNumber(0)
20          .build();
21
22      when(userRepository.findById(credentialsDTO.email()))
23          .thenReturn(Optional.empty());
24      when(encoder.encode(credentialsDTO.password()))
25          .thenReturn("encodedPassword");
26      when(userRepository.save(any(User.class)))
27          .thenReturn(user);
28      when(libraryRepository.save(any(Library.class)))
29          .thenReturn(library);
30      when(jwtService.generateToken(user))
31          .thenReturn("jwtToken");
32
33      UserPayload result = authenticationService.register(
34          credentialsDTO);
35
36      assertEquals(HttpStatus.OK.value(), result.statusCode());
37      assertEquals("jwtToken", result.token());
38      assertEquals(user.getUsername(), result.email());
39      assertEquals(user.getName(), result.username());
40      assertEquals(user.getRole().toString(), result.role());
41      assertEquals(library.getId(), result.libraryId());
42  }
```

5.5 Test ChangePassword() - WhiteBox

```
1  @Test
2  public void testChangePassword_UserNotFound() { // Test user
3      non trovato che quindi mi ritorna un optional vuoto
4      when(userRepository.findById("test@example.com")).thenReturn(Optional.empty());
5      RegistrationDTO registrationDTO = new RegistrationDTO("test@example.com", "oldPassword", "newPassword");
6
7      Payload result = authenticationService.changePw(registrationDTO);
8
9      assertEquals(HttpStatus.CONFLICT.value(), result.statusCode());
10     assertEquals("Impossibile completare la richiesta. I dati specificati non sono presenti nel sistema", result.msg());
11 }
12
13 @Test
14 public void testChangePassword_InvalidOldPassword() { // Test password vecchia invalida per qualche motivo specificato
15     nella funzione
16     when(encoder.matches(anyString(), anyString())).thenReturn(false);
17     RegistrationDTO registrationDTO = new RegistrationDTO("test@example.com", "oldPassword", "newPassword");
18
19     Payload result = authenticationService.changePw(registrationDTO);
20
21     assertEquals(HttpStatus.BAD_REQUEST.value(), result.statusCode());
22     assertEquals("Impossibile completare la richiesta. I dati specificati non sono validi", result.msg());
23 }
24
25 @Test
26 public void testChangePassword_OldEqualsNewPassword() {
27     when(encoder.matches("newPassword", user.getPassword())).thenReturn(true);
28     RegistrationDTO registrationDTO = new RegistrationDTO("test@example.com", "newPassword", "newPassword");
29
30     Payload result = authenticationService.changePw(registrationDTO);
31
32     assertEquals(HttpStatus.BAD_REQUEST.value(), result.statusCode());
33 }
```

```
31         assertEquals("Impossibile completare la richiesta. I dati
32                     specificati non sono validi", result.msg());
33     }
34
35     @Test
36     public void testChangePassword_Success() {
37         when(encoder.matches("oldPassword", user.getPassword()))
38             .thenReturn(true);
39         when(encoder.encode("newPassword")).thenReturn("
40             encodedNewPassword");
41         RegistrationDTO registrationDTO = new RegistrationDTO("test@example.com", "oldPassword", "newPassword");
42
43         Payload result = authenticationService.changePw(
44             registrationDTO);
45
46         assertEquals(HttpStatus.OK.value(), result.statusCode());
47         assertEquals("Operazione completata con successo", result.
48             msg());
49         verify(userRepository, times(1)).save(user);
50     }
51 }
```

6 Valutazione dell'usabilità sul campo

All'interno di quest'ultimo capitolo si analizzerà l'**usabilità sul campo** con un prodotto semi-finito.

Tutto ciò è importante in quanto va ad influenzare direttamente l'esperienza dell'utente, l'efficacia operativa, la qualità del prodotto e la capacità di raggiungere un pubblico più ampio.

Investire sull'usabilità è essenziale per poter creare un prodotto di successo e per poter soddisfare le esigenze dei clienti.

I punti cruciali, infatti, sono:

- **Soddisfazione dell'utente:**

- Un'interfaccia utente intuitiva e facile da utilizzare migliora l'esperienza dell'utente finale e aumenta la sua soddisfazione.
- Gli utenti apprezzano e preferiscono prodotti e applicazioni che sono semplici e piacevoli da utilizzare.

- **Efficienza operativa:**

- Un'interfaccia utente ben progettata consente agli utenti di svolgere i loro compiti in modo più rapido ed efficiente.
- Maggiore produttività e una riduzione dei tempi e degli sforzi necessari per completare le attività.

- **Riduzione degli errori:**

- Un'interfaccia intuitiva e con feedback chiari aiuta a prevenire gli errori degli utenti durante l'utilizzo del prodotto.
- Ciò riduce i costi e il tempo necessario per correggere gli errori e migliorare la qualità complessiva del prodotto.

- **Accessibilità e inclusività:**

- Un'interfaccia utente ben progettata è accessibile a utenti con diverse abilità, età e background.
- Questo aumenta l'inclusività e assicura che il prodotto possa essere utilizzato da un'ampia base di utenti.

- **Adattabilità e flessibilità:**

- Un'interfaccia utente ben progettata è in grado di adattarsi a diversi contesti d'uso, dispositivi e requisiti.
- Ciò aumenta la versatilità e la longevità del prodotto, rendendolo più adattabile alle esigenze future.

6.1 Valutazione dell'applicativo

L'usabilità sul campo è stata valutata grazie a due operazioni in particolare: **valutazioni euristiche** e utilizzo dei **file di log** (trattati in seguito).

Per quanto riguarda la prima, sono stati individuati tre esperti a cui sottoporre alcune domande, tenendo a mente le **8 regole di Schneiderman**.

Tali quesiti hanno come scopo quello di testare l'applicativo e trovare eventuali difetti. Le domande che sono state poste sono le seguenti:

- *C'è un alto grado di usabilità universale?*
- *Ci sono abbastanza riscontri informativi?*
- *Gli errori sono facilmente attivabili? E se si, c'è modo di annullarli?*
- *Il carico di memoria a breve termine è basso?*
- *L'applicazione è grado di rispondere in tempi utili?*
- *E' all'altezza di altri competitors?*

6.1.1 Metodo euristico

Gli utilizzatori individuati hanno avuto piena libertà di esplorare l'app ma sono stati dati loro due compiti⁴ generali da eseguire:

- Primo compito: Creare una playlist ed inserire almeno una canzone al suo interno
- Secondo compito: Cercare un brano musicale ed ascoltarlo

Successivamente alla valutazione euristica gli stessi utenti sono stati valutati per scoprire chi è stato l'utilizzatore più attento e chi meno.

I compiti assegnati sono stati eseguiti tutti in tempi brevi dai nostri valutatori. Si ha, dunque, una misura dell'usabilità abbastanza elevata che va a centrare l'obiettivo iniziale ovvero, quello di avere un grado di usabilità sul campo molto vicino a quello effettuato sui prototipi.

Evitando cambiamenti durante lo sviluppo dell'app che avrebbe portato ad una maggiore quantità di tempo e risorse impiegate, si è evitato il rischio di abbassare il grado di usabilità del prodotto rispetto al prototipo, falsando ed invalidando quest'ultimo.

	Errore creazione playlist	Creazione playlist	Bug aggiunta canzone a playlist	Errore ricerca brano musicale	Bug riproduzione brano da player
Francesco S.		X	X		
Flavia P.		X		X	X
Erika D.		X			

Come rappresentato in tabella i difetti riscontrati dagli utilizzatori sono tre.

NOTA: Abbiamo voluto indicare che la creazione di una playlist sia andata a buon fine per tutti gli utilizzatori in quanto era uno dei compiti che era stato richiesto

⁴abbiamo già fornito agli utilizzatori un profilo utente sull'applicativo, evitando così di dover passare per la registrazione

- **Bug aggiunta canzone a playlist:** Quando l'utente tentava di selezionare una canzone da aggiungere alla playlist, veniva presentata una schermata che mostrava tutte le playlist create sul suo profilo. Tuttavia, in alcuni casi, l'utente non riusciva a selezionare una singola playlist specifica, il che creava problemi nell'aggiunta del brano desiderato alla playlist corretta.

Questo errore è stato risolto

- **Errore ricerca brano musicale:** Se non vengono rispettati i parametri di ricerca sviluppati dall'ingegnere del software responsabile di questa funzionalità, l'applicativo non sarà mai in grado di restituire in modo accurato la traccia musicale desiderata dall'utente finale.
- **Bug riproduzione brano da player:** Se l'utente non concede il permesso per le notifiche quando l'applicazione è in esecuzione, il player multimediale non potrà essere visualizzato nella barra delle notifiche. Di conseguenza, l'utente non sarà in grado di utilizzare il player al di fuori dell'applicazione stessa.

Sono stati ritenuti importanti tutti gli errori che sono stati riscontrati dagli osservatori ma, in particolare, è stata Flavia P. ad essere stata individuata come *utente più attento*, in quanto è riuscita a riscontrare due problematiche che gli sviluppatori dell'applicativo risolveranno in un futuro immediato. Gli utilizzatori sono stati invitati a rispondere alle domande citate ad inizio paragrafo, delle quali sono state riportate le loro risposte.



Francesco S.



Posso affermare che l'app è abbastanza chiara ed esplicativa



I risconti informativi che ho incontrato sono stati prettamente legati ad alcune funzioni, avrei preferito averne qualcuno in più



Fortunatamente, non ho riscontrato molti errori grazie, anche, ad un ottimo lavoro di controllo svolto. Purtroppo, l'unico che ho incontrato è stato difficile da annullare



Molto basso, devo ammettere



Si, ci riesce tranquillamente



Con l'inserimento di nuove funzionalità, sono sicuro che possa diventarlo



Flavia P.



Mh si, abbastanza



Sembra di sì



Gli errori che ho riscontrato sono stati facilmente attivabili. Uno di essi è anche facile da risolvere ma per quanto riguarda il secondo, bhe, no



Basso, sì



Assolutamente sì



Credo di sì

 Erika D.

Grazie alle interviste condotte, il team di sviluppo ritiene di aver ottenuto risultati soddisfacenti. Alcuni dei problemi e degli errori riscontrati sono stati già risolti, mentre per altre questioni il team si impegnerà a trovare una soluzione il prima possibile.

Le risposte ricevute dagli utenti sono state nel complesso positive. Ciò sembra essere dovuto in parte alla fiducia accordata agli utilizzatori e ai controlli nonché alle conferme richieste durante le operazioni più critiche all'interno dell'applicazione.

Questi accorgimenti hanno contribuito a migliorare l'affidabilità percepita del sistema.

6.1.2 Metodo con monitoraggio ed analisi dei log

Debug library in Android

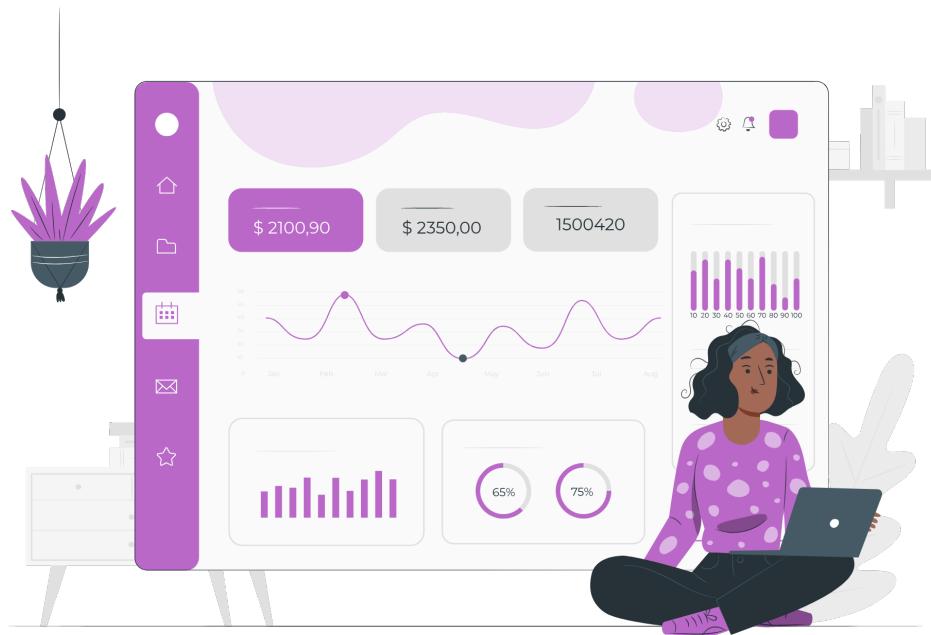
Per utilizzare un metodo facile e intuitivo di debuggare il codice android durante lo sviluppo è stata ritenuta valida la **libreria Log** fornita da Android Studio; i log vengono visualizzati all'interno del **Logcat** dell'applicativo:

```
Log.d( tag: "MainActivity",  msg: "onCreate called");
```

```
Log.d(TAG,  msg: "UTENTE:" + email);  
Log.d(TAG,  msg: "UTENTE:" + password);
```

```
Log.d(TAG,  msg: "Numero di fragment nello stack: " + getSupportFragmentManager().getBackStackEntryCount());
```

2024-06-07 23:55:14.862 14756-14756 MainActivity	com.example.soundlab	D onCreate called
2024-06-07 23:55:15.003 14756-14756 MAIN	com.example.soundlab	D UTENTE:string@string.com
2024-06-07 23:55:15.003 14756-14756 MAIN	com.example.soundlab	D UTENTE:string



Analisi dei file di log by Google Analytics

Grazie all'uso di Firebase è stato possibile raccogliere un gran numero di informazioni circa il gradimento dell'applicazione e le sue prestazioni.

Si è fatto uso di:

- **Analytics:** per la raccolta di dati e statistiche sugli eventi all'interno dell'applicazione
- **Performance:** per la raccolta di dati circa i tempi di risposta e di avvio

7 Conclusione e valutazione del team

In questo progetto abbiamo dato vita a **SoundLab**, una piattaforma che permette agli utenti di ascoltare e gestire la propria musica preferita in modo intuitivo e personalizzato.

Attraverso l'analisi dei requisiti, la progettazione del modello funzionale e architetturale, e l'implementazione delle principali funzionalità, siamo riusciti a realizzare un'applicazione che risponde alle esigenze degli utenti target identificati.

Durante lo sviluppo, abbiamo posto particolare attenzione all'esperienza utente, creando un'interfaccia grafica semplice e intuitiva che facilita la navigazione e l'interazione con le varie funzionalità. Inoltre, l'adozione di tecnologie moderne e l'architettura modulare garantiscono flessibilità e scalabilità del sistema.

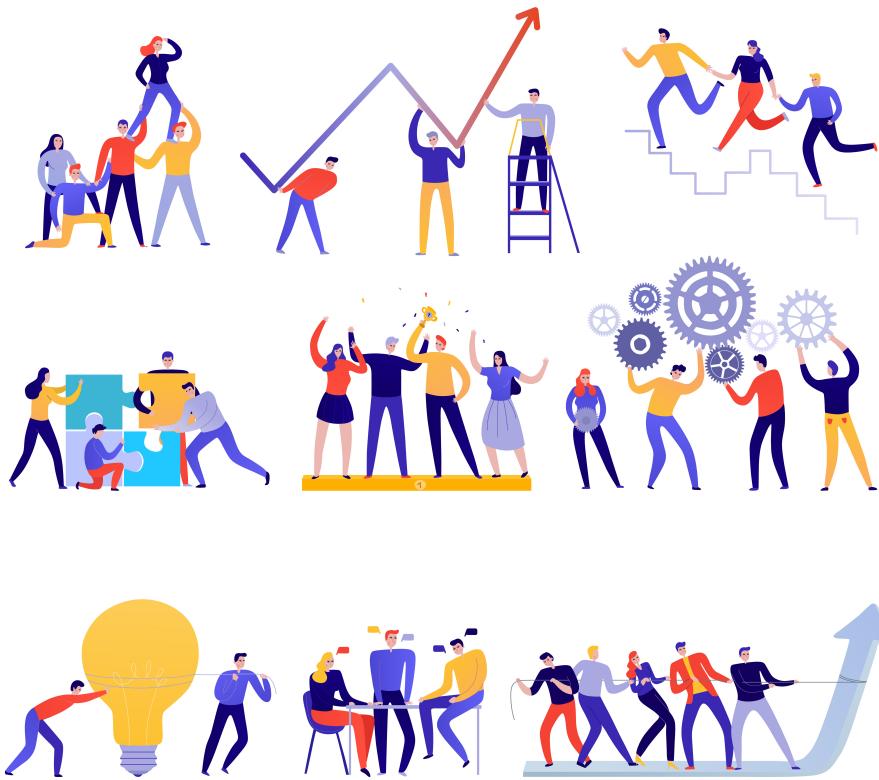
Nonostante i risultati raggiunti, esistono ancora numerosi margini di miglioramento e sviluppi futuri per l'applicazione.

Il team pensava a tali futuri sviluppi:

- Integrazione di funzionalità avanzate di analisi dei dati e raccomandazione di brani e playlist in base alle preferenze dell'utente
- Implementazione di funzionalità collaborative, come la condivisione di playlist tra utenti
- Integrazione di funzionalità di creazione e modifica di brani musicali
- Implementazione di meccanismi di monetizzazione, come abbonamenti premium o vendita di contenuti aggiuntivi

Questi rappresentano solo alcuni degli ambiti di miglioramento e sviluppo futuro che potranno rendere SoundLab un'applicazione sempre più completa e in grado di soddisfare le esigenze degli appassionati di musica.

Siamo estremamente soddisfatti del risultato finale del nostro lavoro. Il prodotto che abbiamo realizzato è non solo funzionale e rispondente alle esigenze degli utenti, ma anche innovativo e ben progettato.



Abbiamo lavorato sodo come team per raggiungere questo obiettivo. È stato gratificante vedere il nostro lavoro prendere forma e concretizzarsi. Abbiamo affrontato diverse sfide lungo il percorso, ma grazie alla **collaborazione** e alla tenacia di tutto il team siamo riusciti a superarle. Siamo fiduciosi che il prodotto avrà un grande successo e avrà un impatto positivo sulla vita degli utenti; siamo davvero tanto entusiasti di vedere come verrà utilizzato e di come si evolverà nel tempo. Ci teniamo a specificare di quanto sia stato **stimolante** lavorare in questo team, in quanto è stata fondamentale la **collaborazione** tra noi membri. Insieme abbiamo affrontato sfide complesse, discusso diverse soluzioni e trovato il modo migliore per procedere. Nonostante avessimo opinioni e competenze diverse all'interno del gruppo, ciò ha portato a un processo decisionale più ricco e a soluzioni più innovative. Oltre al lavoro di squadra, riteniamo che sia stato molto formativo l'essere parte di questo team. **Abbiamo imparato tantissimo l'uno dagli altri**, sia per quanto riguarda le competenze tecniche che l'approccio al lavoro, delineato da tantissima **serietà e professionalità**. Le nostre conoscenze si sono ampliate e sono state sviluppate nuove abilità attraverso la formazione interna e la partecipazione a chiamate di "lavoro" e di "brainstorming". In definitiva, la nostra esperienza è da valutare come positiva.



© SoundLab è un prodotto ideato e sviluppato da poveri studenti di informatica. Gli autori, in quanto tali, detengono ogni diritto dello stesso in maniera esclusiva.