

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Tesi di laurea di I livello in  
**Informatica**

**Sviluppo di un'applicazione Android  
per la rilevazione e la gestione del  
plagio musicale**

Relatori:

**Rocco Zaccagnino**  
**Roberto De Prisco**

Candidato:

**Antonio Giometta**  
**Mat. 05121 06027**

ANNO ACCADEMICO 2020/2021

*These are the days we've been waiting for  
On days like these, who could ask for more?  
Keep them coming 'cause we're not done yet  
These are the days we won't regret  
These are the days we won't forget*

*Live a life, you will remember*

*Tim Bergling, Avicii*

---

# INDICE

<b>1 ABSTRACT</b>	<b>5</b>
<b>2 INTRODUZIONE</b>	<b>7</b>
2.1 Plagio . . . . .	7
2.1.1 Definizione di Plagio . . . . .	7
2.1.2 Esempi di Plagio Musicale . . . . .	8
2.2 Problema . . . . .	8
2.2.1 Tecniche Esistenti . . . . .	8
2.2.2 Ambito Legislativo . . . . .	9
2.2.3 Obiettivo della Tesi . . . . .	9
2.2.4 Organizzazione della Tesi . . . . .	10
<b>3 FASE DI RICERCA</b>	<b>12</b>
3.1 Google - Youtube Content ID . . . . .	13
3.2 Fraunhofer IDMT . . . . .	13
3.3 Text-Similarity . . . . .	14
3.3.1 Algoritmi String-Based . . . . .	15
3.3.2 Algoritmi Corpus-Based . . . . .	15
3.3.3 Algoritmi Knowledge-Based . . . . .	15

---

## INDICE

---

<b>4 FASE DI STUDIO TECNOLOGICO</b>	<b>16</b>
4.1 Funzionalità legacy riutilizzate . . . . .	16
4.1.1 Metriche utilizzate nel sistema legacy . . . . .	16
4.1.2 Maas - Music as String . . . . .	18
4.1.3 Ensemble methods e modello di Clustering . . . . .	18
4.2 Python . . . . .	19
4.3 Flask . . . . .	20
4.3.1 Flask-SQLAlchemy . . . . .	20
4.3.2 Flask-Mail . . . . .	21
4.4 Java per Android . . . . .	21
4.5 XML . . . . .	21
4.6 JSON . . . . .	22
<b>5 FASE DI PROGETTAZIONE</b>	<b>24</b>
5.1 Metodologia e materiali utilizzati . . . . .	24
5.2 Design Pattern MVC . . . . .	24
5.2.1 Oggetti Model . . . . .	25
5.2.2 Oggetti View . . . . .	26
5.2.3 Oggetti Controller . . . . .	26
5.3 Architettura del Sistema . . . . .	27
5.3.1 Sistema Client-Server . . . . .	27
5.3.2 SubSystem Main . . . . .	28
5.3.3 SubSystem Sentences . . . . .	28
5.3.4 SubSystem Users . . . . .	29
5.4 Tipologie di utenti . . . . .	29
<b>6 FASE DI SVILUPPO</b>	<b>31</b>
6.1 Tecnologie di implementazione . . . . .	31
6.1.1 Activity e Fragment . . . . .	31
6.1.2 Dataset, Ensemble e Clustering . . . . .	32
6.1.3 Flask Blueprint . . . . .	32
6.1.4 Models . . . . .	33

---

---

## INDICE

---

6.1.5	Routes . . . . .	34
6.1.6	Icone utilizzate nei casi di plagio . . . . .	34
6.2	IDE Utilizzate . . . . .	35
6.2.1	PyCharm . . . . .	35
6.2.2	Android Studio . . . . .	35
6.3	Fasi di implementazione . . . . .	36
6.3.1	Definizione della struttura del progetto . . . . .	36
6.3.2	Permessi Android . . . . .	36
6.3.3	Inizializzazione e configurazione del progetto . . . . .	37
6.3.4	Gestione della connessione al server legacy . . . . .	37
6.3.5	Sviluppo delle funzionalità . . . . .	38
6.4	Menu . . . . .	39
6.5	SubSystem Main . . . . .	39
6.5.1	Caricamento brani . . . . .	39
6.5.2	Calcolo dei valori del confronto . . . . .	40
6.5.3	Salvataggio dei dati del confronto corrente . . . . .	41
6.5.4	Visualizzazione e schematizzazione dei risultati . . . . .	41
6.6	SubSystem Users . . . . .	43
6.6.1	Autenticazione . . . . .	43
6.6.2	Visualizzazione e Aggiornamento informazione dell'account . . . . .	45
6.7	SubSystem Sentences . . . . .	48
6.7.1	Visualizzazione casi di plagio . . . . .	48
6.7.2	Visualizzazione singolo caso di plagio . . . . .	51
6.7.3	Ricerca casi di plagio . . . . .	53
6.7.4	Inserimento caso di plagio . . . . .	53
6.7.5	Aggiornamento dataset Ensemble e Clustering . . . . .	55
6.8	Sviluppi futuri . . . . .	55
7	RINGRAZIAMENTI	57
	BIBLIOGRAFIA	59

---

---

---

# CAPITOLO 1

---

## ABSTRACT

Il plagio musicale, *l'atto di copiare un brano o parte di esso da altri compositori*, è diventato un problema estremamente rilevante a causa degli elevati investimenti nel mercato musicale.

Esistono vari aspetti di un brano che si possono considerare; in linea generale, nell'ambito della musica leggera, si considera elemento distintivo di un brano la linea melodica: è ad essa, e non al timbro e agli accordi, che si guarda per verificare se vi è stato plagio o meno.

Dal punto di vista legislativo non sono presenti metodologie chiare ed oggettive utilizzabili per la valutazione, spesso si fa ricorso ad un consulente tecnico.

La nostra applicazione si propone di fornire supporto alle scelte decisionali di un giudice, offrendo metriche oggettive di valutazione ed una chiara visualizzazione dei risultati. In questo lavoro di tesi ci occupiamo quindi della creazione di una applicazione Android per la rilevazione del plagio tra due brani, integrando al suo interno una metaeuristică adattiva basata su text-similarity e Clustering; fornendo inoltre funzionalità di inserimento e gestione di plagio all'interno della base dati.

Il mio lavoro di tesi si basa sul problema di dover far comunicare una piattaforma legacy esistente con una applicazione Android, e più in generale,

---

## 1. ABSTRACT

---

su tutta la realizzazione del back-end.

La realizzazione dell'applicazione si basa principalmente sulle tecnologie Python Flask, Java per Android, XML, JSON.

Come risultato abbiamo ottenuto **una delle prime applicazioni per la gestione e la rilevazione di casi di plagio.**

---

---

# CAPITOLO 2

---

## INTRODUZIONE

### 2.1 Plagio

#### 2.1.1 Definizione di Plagio

Con il termine plagio, nell'ambito del diritto d'autore, si identifica "l'appropriazione, tramite copia totale o parziale, della paternità di un'opera prodotta dall'ingegno altrui". Il plagio può avere diverse forme a seconda dell'ambito in cui ci si trova; in particolare, le due macroaree di riferimento per il plagio sono le seguenti:

- Il plagio letterario, relativo all'utilizzo di parti di una pubblicazione o di uno scritto, sia esso un libro, una poesia o altri tipi di opera letteraria, appropriandosi in modo fraudolento della proprietà dell'opera.
- Il plagio musicale, ovvero l'atto di copiare un brano o parte di esso da altri compositori, il quale è diventato un fenomeno estremamente importante e oggetto di studi, soprattutto a causa degli elevati investimenti nel mercato musicale.

Nell'ambito di questa tesi ci focalizzeremo sul plagio musicale.

### 2.1.2 Esempi di Plagio Musicale

Un caso eclatante di plagio musicale fu quello di “Albano contro Michael Jackson” [1] dove, nel 1992, Albano fece causa a Michael Jackson poiché riteneva che ”Will you be there” fosse copiata da ”I cigni di Balaka”, un suo brano rilasciato nel 1987. Inizialmente Albano vinse la causa, poiché la giuria stabilì che il plagio sussistesse (a determinare il plagio contribuì il Maestro Ennio Morricone). Tuttavia, in un secondo momento, fu dimostrato che entrambe le canzoni fossero ispirate a ”Bless You For Being An Angel” degli Ink Spots. Quindi, se prima fu stabilito che Jackson dovesse pagare quattro miliardi di lire, alla fine fu lo stesso Albano a farsi onere delle spese del processo.

Un altro caso famoso fu il caso “Harrison contro Chiffons” [2], dove il cantante ed ex componente dei Beatles fu accusato di aver copiato parte della canzone ”He’s So Fine” adattandola in ”My Sweet Lord”. Alla fine il giudice decretò che Harrison avesse involontariamente plagiato ”He’s So Fine” e fu costretto a dare alcuni dei proventi del suo album, per un totale di 1.6 milioni di dollari, alle Chiffons.

## 2.2 Problema

### 2.2.1 Tecniche Esistenti

A discapito del forte interesse economico relativo ai casi di plagio, le tecniche esistenti per la rilevazione del plagio musicale sono semplici. Ad esempio, YouTube utilizza una tecnologia di identificazione per i video disponibili sulla propria piattaforma, definita ”Content ID” [3]. La tecnologia consiste nell’utilizzare tools di matching audio e video per verificare la corrispondenza tra un contenuto caricato dai vari utenti e l’audio e video forniti dai possessori del contenuto originale. Nel caso YouTube trovasse un video che utilizza materiale registrato sotto CopyRight, per conto del proprietario originale possono essere messe in atto, anche in modo automatico azioni come

## **2. INTRODUZIONE**

---

ad esempio il blocco del video, monetizzazione, ecc. Per quanto questo metodo possa essere ottimale per confrontare contenuti identici tra loro, esso non è sufficiente però, a riconoscere un plagio musicale, poiché due brani potrebbero essere differenti, ma potrebbe sussistere tra di loro un plagio, sia esso relativo alla melodia o al testo dell'opera.

### **2.2.2 Ambito Legislativo**

In ambito legislativo, inoltre, non è molto chiaro come rilevare un plagio tra due brani. Per le composizioni musicali non esiste una regola generale in base alla quale un numero minimo di note, o di battute, uguali tra due opere configura il plagio [4]. La giurisprudenza ha infatti affermato: La parziale assonanza tra due composizioni musicali, casuale e limitata a poche battute, esclude che tra esse vi sia plagio, soprattutto quando esse si ispirano a diverse tradizioni musicali [5]. Nella pratica, però, basta che un brano susciti nell'ascoltatore il riconoscimento di un pezzo coperto da diritto d'autore ad esso antecedente per supporre un plagio. Per procedere giuridicamente, un giudice nomina un CTU (consulente tecnico d'ufficio) per redigere una perizia giurata, al quale viene proposto l'ascolto dei due brani (l'originale e l'eventuale plagio). Se il giudice riconosce le ragioni dell'attore (colui che intraprende l'azione legale), l'autore del plagio rischia il ritiro del pezzo dal mercato con sanzioni pecuniarie, oppure il riconoscimento all'autore originale di parte dei diritti (con relative royalties) su quel pezzo [6].

Nell'ambito della musica leggera, in linea generale, l'elemento distintivo di un brano è la linea melodica. Infatti, tale linea melodica risulta essere l'aspetto predominante nell'analisi svolta per verificare se sussista un caso di plagio o meno [7]

### **2.2.3 Obiettivo della Tesi**

L'obiettivo di questa tesi è lo sviluppo di una applicazione Android per supportare le decisioni di un giudice in un caso giudiziario di presunto plagio,

## 2. INTRODUZIONE

---

o di un artista che voglia confrontare il suo brano con altri che ritiene possano essere in plagio con la propria opera, o anche per evitare un proprio plagio involontario, come accadde nel caso di George Harrison. La piattaforma legacy esistente offre funzionalità di calcolo, di inserimento, di ricerca ed in generale di tipo gestionale dei singoli casi di plagio per poter visualizzare, rispettando i dogmi della visual analytics<sup>1</sup>, i singoli casi di plagio e poter così procedere con decisioni ponderate. Tale piattaforma legacy, infatti, permette di calcolare i dati del confronto.

Sfruttando le funzionalità della piattaforma legacy, l'oggetto del lavoro di tesi si basa sul rendere fruibili tali servizi, mediante App dedicata, su dispositivi MobileAndroid, in particolare il problema di dover far comunicare una piattaforma legacy esistente con una applicazione Android, e più in generale, su tutta la realizzazione del back-end.

### 2.2.4 Organizzazione della Tesi

La struttura della tesi rispecchia i passi affrontati per progettare ed elaborare la piattaforma:

- **Fase di Ricerca**, durante la quale sono stati raccolti dati relativi al plagio, specialmente in ambito musicale, nonché delle possibili tecnologie implementative da adottare per lo sviluppo del nostro progetto, nonché la ricerca delle soluzioni disponibili sul mercato allo stato d'arte.
- **Fase di Studio Tecnologico**, nella quale sono state approfondite le tecnologie e i linguaggi necessari al progetto, come Java per Android, XML, JSON, Python, Flask ed i suoi moduli nonché gli algoritmi e le funzionalità di progetti preesistenti da cui partire, per utilizzarle all'interno della nostra applicazione.

---

<sup>1</sup>Tradotto dall'inglese, l'analisi visiva è una branca dei campi della visualizzazione delle informazioni e della visualizzazione scientifica che si concentra sul ragionamento analitico facilitato da interfacce visive interattive

## 2. INTRODUZIONE

---

- **Fase di Progettazione**, nella quale sono state valutate le funzionalità offerte dal sistema legacy, e la sua struttura. La progettazione si basa sulla creazione di un'applicazione client Android in grado di poter fruire dei servizi presenti sul server legacy, con la minima modifica possibile di quest'ultimo.
- **Fase di Sviluppo**, durante la quale si è messo in atto quanto definito in fase di progettazione, creando un'applicazione Android in grado di interfacciarsi con il server legacy.

---

---

## CAPITOLO 3

---

### FASE DI RICERCA

La Fase di Ricerca può essere suddivisa in tre sottofasi:

- Nella prima sottofase abbiamo ricercato informazioni sulla definizione del plagio e del plagio musicale, e di come la legislazione trattasse questa problematica.

Come già detto nell'introduzione, non ci sono metodologie chiare utilizzate dalla legislatura, ma solo metodi generici di risoluzione della questione. Dalle ricerche effettuate sono emersi come risultati validi esclusivamente le tecnologie *"Youtube Content ID"* di Google e *"IDMT"* della Fraunhofer.

- Nella seconda sottofase ci siamo focalizzati sugli algoritmi e le metriche disponibili allo stato dell'arte utilizzate dal sistema legacy. La maggior parte delle attività è stata dedicata agli algoritmi di Text-Similiraty.
- Nella terza sottofase ci siamo focalizzati sulla ricerca di applicazione android, disponibili allo stato dell'arte, che permettessero il rilevamento del plagio musicale, **senza però rilevare alcuna soluzione disponibile sul mercato.**

## 3.1 Google - Youtube Content ID

Nelle ricerche di sistemi di rilevazione del plagio abbiamo trovato interessante *Content ID*, un sistema di identificazione e rilevazione dei propri contenuti su Youtube, infatti *Content ID* consente ai titolari del copyright di identificare e gestire facilmente i propri contenuti.

I video caricati su Youtube vengono esaminati e confrontati con un database di file che Youtube ha ricevuto dai proprietari dei contenuti.

Spetta al titolare del copyright decidere cosa fare nel caso in cui i contenuti di un video di Youtube corrispondono a una delle sue opere. Quando viene trovata una corrispondenza, il video riceve una rivendicazione di *Content ID*. I titolari del copyright possono intraprendere diverse azioni nei confronti di contenuti corrispondenti ai propri:

- Bloccare la visione dell'intero video
- Monetizzare il video pubblicando annunci; talvolta condividendo le entrate con l'utente che ha caricato il video
- Tracciare le statistiche sulle visualizzazioni del video

Come già detto però, questo sistema è capace di rilevare, in ambito musicale, un plagio solo se il brano, nella sua interezza, viene caricato come proprio o senza permessi.

## 3.2 Fraunhofer IDMT

Un altro interessante strumento è il *Fraunhofer IDMT*, un software di rilevazione del plagio che permette agli utenti di rilevare sia un plagio melodico sia quello di campioni nei brani musicali.

Il software di rilevamento del plagio *IDMT* offre vari strumenti per supportare la valutazione oggettiva di presunti casi di plagio musicale.

Permette anche la scansione di nuove registrazioni musicali per riconoscere l'utilizzo di campioni individuali di vecchie registrazioni.

### 3. FASE DI RICERCA

---

Campioni ricorrenti possono essere rilevati anche se utilizzati a diverse velocità o frequenze, o se miscelati con ulteriori tracce musicali.

Dai risultati analizzati è poi possibile rimuovere il campione rilevato dalla traccia musicale, in questa maniera il software si propone di poter provare la presenza o meno del plagio in maniera intuitiva ed affidabile.

Gli strumenti del software permettono quindi di:

- Comparare due brani musicali in termini di temi melodici, o in termini di motivi simili o identici.
- Utilizzare algoritmi specializzati per valutare automaticamente il grado di similarità tra due sequenze sotto analisi, per permettere una misura oggettiva in caso di plagio rilevato[8].

Nonostante gli interessanti spunti sul plagio musicale, il software non da però maggiori riferimenti agli algoritmi ed alle metriche utilizzate.

### 3.3 Text-Similarity

Nel web sono disponibili molte metriche per quanto riguarda il plagio testuale, alcune delle quali utilizzate come parte integrante delle funzionalità del sistema legacy, applicandole a rappresentazioni di brani musicali.

In particolare applicheremo la "*Text-Similarity*" [9]

Le metriche di text-similarity giocano un ruolo sempre più importante in ricerche e applicazioni relative ai testi, in task quali il recupero delle informazioni, la classificazione dei testi, clustering di documenti, rilevazione e tracciamento degli argomenti, generazione di domande, ecc.

Trovare similarità tra le parole è una parte fondamentale della text similarity, successivamente utilizzata come parte prima per similarità tra fasi, paragrafi ed interi documenti.

Le parole possono essere simili in due maniere:

- **lessicale:** le parole sono simili lessicalmente se hanno una sequenza di caratteri simili.

- **semantica:** le parole sono simili semanticamente se sono la stessa cosa, sono l'opposto l'una dell'altra, sono utilizzate nello stesso modo o contesto, oppure se una è un tipo dell'altra.

La similarità lessicale è rilevata attraverso diversi algoritmi *String-Based*, mentre la similarità semantica viene rilevata da algoritmi *Corpus-Based* e *Knowledge-Based*.

#### 3.3.1 Algoritmi String-Based

Gli algoritmi String-Based operano su sequenze di stringhe e composizione dei caratteri. Una String metric è una metrica che misura similarità o dissimilarità tra due stringhe di testo per una corrispondenza o comparazione approssimativa tra le stringhe; un esempio potrebbe essere la **distanza di edit**, che tra due stringhe A e B rappresenta il numero minimo di modifiche elementari che consentono di trasformare la A nella B [10].

#### 3.3.2 Algoritmi Corpus-Based

La similarità Carpus-Based è una misura semantica che determina la similarità tra parole in base ad informazioni ottenute da grosse corpora<sup>1</sup>. Alcuni esempi di algoritmi corpus-based sono **HAL,LSA**, ed **ESA**.

#### 3.3.3 Algoritmi Knowledge-Based

La similarità Knowledge-Based è una misura di similarità semantica che determina il grado di similarità tra parole sfruttanto informazioni derivate da network semantici. Un esempio di network semantico è **WordNet** [11].

---

<sup>1</sup>Le corpora sono collezioni, per lo più di grandi dimensioni, di testi orali o scritti prodotti in contesti comunicativi reali.

---

---

# CAPITOLO 4

---

## FASE DI STUDIO TECNOLOGICO

In questo capitolo discuteremo delle funzionalità estrapolate dal progetto legacy<sup>1</sup> per la rilevazione del plagio e le diverse tecnologie adottate per lo sviluppo dell'applicazione Android, la gestione e visualizzazione delle informazioni e i moduli e le librerie utilizzate.

### 4.1 Funzionalità legacy riutilizzate

Dai lavori precedenti abbiamo estrapolato diverse funzionalità, adattando e aggioranndo le stesse per i nostri scopi.

#### 4.1.1 Metriche utilizzate nel sistema legacy

Le metriche utilizzate nel sistema legacy sono metriche di similarità String-Based (ovvero **character based** e **term based**) e sono:

- Jaccard Similarity (JS)
- Cosine Similarity (CS)

---

<sup>1</sup>Per "Sistema Legacy" o "Progetto Legacy" ci riferiamo alla precedente applicazione, basata sui lavori di L.Ciaravola [20], a loro volta basata sul lavoro di R.Ferraioli ed A.Del Gaizo.[12][18]

## 4. FASE DI STUDIO TECNOLOGICO

---

- Dice's Coefficient (DC)
- Jaro Similarity (JRS)
- Overlap Coefficient (OC)

Di seguito illustriamo brevemente ciascuna metrica utilizzata nel sistema legacy, omettendo le rappresentazioni matematiche poichè già analizzate, descritte e implementate nei lavori precedenti [12]:

- **Jaccard Similarity**

La similarità di Jaccard è utilizzata per determinare quanto due testi si avvicinano tra loro, ovvero quante parole in comune esistono sul totale delle parole.

Il suo range di valori va da 0 ad 1. Dove 1 indica che i due documenti sono identici e ' che i due documenti non si somigliano per nulla.

- **Cosine Similarity**

Anch'essa misura la somiglianza del testo tra due documenti, indipendentemente dalla loro dimensioni. Una parola è rappresentata in uno spazio vettoriale n-dimensionale proiettati in uno spazio multidimensionale. Il suo range di calori va da 0 a 1, dove 1 significa che i due vettori hanno lo stesso orientamento, e 0 indica che i due vettori non hanno somiglianza.

- **Dice's Coefficient**

Noto anche come indice Soresen-Dice, è uno strumento statistico che misura la somiglianza tra due dati. Esso è definito come il doppio del numero di termini comuni nelle stringhe confrontate, diviso per il numero totale di termini in entrambe le stringhe. Anche in questo caso il range di valori varia da 0 a 1; dove 1 significa che le stringhe sono uguali e 0 che le due stringhe sono diverse.

- **Jaro Similarity**

Jaro si basa sul numero e ordine dei caratteri in comune alle due stringhe,

tenendo conto anche delle parole scritte in maniera errata. Il valore della Jaro Similarity varia da 0 a 1, dove 1 significa che le stringhe sono uguali e 0 significa che non è riscontrata alcuna somiglianza tra le due stringhe.

- **Overlap Coefficient**

Simile al Dice's Coefficient, considera due stringhe come una corrispondenza completa, ossia se una è un sottoinsieme dell'altra. I valori, come per gli altri algoritmi, variano tra 0 e 1, dove 1 significa che le due stringhe sono uguali, e 0 significa che non è riscontrata alcuna somiglianza tra le due stringhe.

### 4.1.2 Maas - Music as String

Le funzionalità MaaS ("Music as String") del progetto legacy, utilizzate per la conversione di brani XML in una stringa testuale utilizzata per la rilevazione del plagio.

### 4.1.3 Ensemble methods e modello di Clustering

Il metodo *Ensemble* trae ispirazione da tecniche che creano più modelli deboli, combinandoli insieme per produrre risultati migliori, utilizza quindi algoritmi di text similarity quali:

- Jaccard Similarity
- Cosine Similarity
- Dice's Coefficient
- Jaro Similarity
- Overlap Coefficient

Questi algoritmi verranno combinati in coppie o triple di metriche di text-similarity ogni volta. Definisce poi un threshold comune per tutte le metriche, sfruttando l'ottimizzazione **Pareto Front** [13], che se superata

permette di dichiarare la sussistenza di plagio tra due brani.

Successivamente, se necessario, si combinano i risultati delle metriche di text-similarity e la rappresentazione testuale dei brani con un modello di Clustering basato su di una rappresentazione vector-based dei brani.

L'insieme degli ***Ensemble Methods*** e del **Modello di Clustering** forma la **MetaEuristica multi-obiettivo adattiva ensemble-based** utilizzata dal sistema legacy.

(Le metriche, i loro valori, il threshold e l'utilizzo del Clustering saranno poi visualizzabili nella nostra applicazione Android tra i risultati calcolati dal server legacy).

## 4.2 Python

Dal momento che il sistema legacy esistente è basato su una piattaforma server Python, e nell'ottica di rendere fruibile i suoi servizi su applicazione Android con il minor numero di modifiche possibili, abbiamo deciso di lavorare direttamente con questo linguaggio.

Python è poi molto completo, con la sua filosofia “tutto compreso” relativa alle librerie.

L'idea è che, installando Python, si ha a disposizione tutto il necessario per lavorare, senza perdere tempo in installazioni supplementari. La libreria standard di Python contiene moduli per lavorare con email, pagine web, database, chiamate al sistema operativo, sviluppo di interfacce utenti grafiche e altro.

Python è anche multipiattaforma; siccome è un linguaggio interpretato, il codice funziona ovunque si trovi un interprete Python, e quasi tutte le piattaforme più usate ne possiedono uno. Esistono versioni di Python per Java (Jython) e .NET (IronPython), per esempio.

### 4.3 Flask

Flask è un micro framework web scritto, per l'appunto, in Python.

È classificato come micro framework poiché non richiede particolari strumenti o librerie [14].

Non ha nessun layer di astrazione per il database, validazione form o nessun altro componente dove librerie di terze parti preesistenti forniscono funzioni comuni. Ad ogni modo, Flask supporta estensioni che possono aggiungere funzionalità applicative come se implementate in Flask stesso. Oltre tutto implementa diverse funzionalità utili, come ad esempio i messaggi Flash per fornire una notifica all'utente.

Esistono estensioni per object-relational mappers, validazione dei form, gestione dell'upload, varie tecnologie aperte d'autenticazione e diversi strumenti relativi a framework comuni [15].

Alcune di queste estensioni sono, per l'appunto, utilizzate nella modifica del sistema legacy. Applicazioni di nota che utilizzano Flask come framework includono Pinterest e LinkedIn [16][17].

Alcuni dei moduli (estensioni di Flask, ma non solo) utilizzati nelle modifiche al server legacy, sono:

#### 4.3.1 Flask-SQLAlchemy

Flask-SQLAlchemy è un'estensione per Flask che aggiunge il supporto per SQLAlchemy all'applicazione. Semplifica l'utilizzo di SQLAlchemy con Flask fornendo diverse funzionalità di supporto per semplificare task comuni. All'interno del nostro progetto è utilizzato per la gestione del database direttamente da codice, sfruttando la molitudine di funzionalità offerte: dal poter creare oggetti derivati da db.Model (che verranno trasformati in tabelle nel database alla sua creazione) fino alla gestione e filtro di tutti i contenuti del database, utilizzando i metodi forniti.

### 4.3.2 Flask-Mail

Flask-Mail è un'estensione Flask che fornisce una semplice interfaccia per il setting SMTP con una applicazione Flask, permettendo l'invio di messaggi dalle Views e dagli script.

All'interno del progetto è utilizzata per l'invio di email di reset password, su richiesta dall'utente.

## 4.4 Java per Android

Esistono due linguaggi di programmazione, Java e Kotlin, che permettono di sviluppare app per Android.

Scegliere il linguaggio è difficile poichè non esiste un linguaggio che sia la risposta giusta a tutti i possibili problemi. La nostra scelta è ricaduta su Java, dal momento che è stato oggetto di corso di studi, e che, per padronanza e per funzioni a noi necessarie, si avvicina di più ai nostri bisogni. Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica, che si appoggia sull'omonima piattaforma software di esecuzione, specificamente progettato per essere il più possibile indipendente dalla piattaforma hardware di esecuzione, tramite compilazione in bytecode prima e interpretazione poi da parte di una Java Virtual Machine.

## 4.5 XML

XML (sigla di eXtensible Markup Language) è un metalinguaggio per la definizione di linguaggi di markup, ovvero un linguaggio basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.

All'interno del nostro progetto, XML viene utilizzato per la stesura dei layout delle varie activity e dei vari fragment.

In XML le informazioni specifiche di un'applicazione sono contenute all'interno di "tag", marcati da parentesi `< >`, che descrivono il contenuto di un documento.

Ogni tag definisce un tipo di elemento e, delimitando con tag ogni singolo dato, siamo in grado di comprenderne la struttura anche se non conosciamo l'applicazione che l'ha generata. Essendo poi i dati autodescrittivi, anche i partner saranno in grado di comprenderli ed elaborarli. Inoltre essi possono essere gestiti anche in futuro quando le applicazioni che li hanno generati saranno diventate obsolete.

L'estensibilità è un'altra caratteristica vincente di XML, in quanto è possibile per i programmati riutilizzare i documenti XML esistenti semplicemente estendendoli con nuovi tag, lasciando che gli elementi chiave del documento originale rimangano comprensibili da tutti gli utilizzatori.

### 4.6 JSON

JSON, acronimo di JavaScript Object Notation, è un formato adatto all'interscambio di dati fra applicazioni client/server.

JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmati di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.

JSON è basato su due strutture:

- Un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.
- Un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Queste sono strutture di dati universali. Virtualmente tutti i linguaggi di programmazione moderni li supportano in entrambe le forme. E' sensato che un formato di dati che è interscambiabile con linguaggi di programmazione debba essere basato su queste strutture.

---

#### **4. FASE DI STUDIO TECNOLOGICO**

---

Questo linguaggio è stato usato per permettere il passaggio di dati tra il server Flask e l'applicazione Android.

---

---

# CAPITOLO 5

---

## FASE DI PROGETTAZIONE

In questa fase illustreremo le metodologie ed i pattern utilizzati nel server legacy e nell'applicazione Android, per poi descriverne le funzionalità.

### 5.1 Metodologia e materiali utilizzati

Le funzionalità di rilevazione del plagio della nostra applicazione si basano sui tre lavori precedenti [12][18][20] basati sulla conversione dei brani musicali in una rappresentazione vector-based, per poi sfruttare le tecniche di Ensemble, utilizzando una combinazione di tecniche di text-similarity [2.1.2], e di Clustering al fine di valutare la presenza di plagio tra i due brani. E la piattaforma web che offre servizi come: gestione, salvataggio e visualizzazione dei casi di plagio, risultato delle funzionalità sopra indicate.<sup>77</sup> Le vecchie funzionalità sono state modificate in modo da poter interagire con la nostra applicazione Android.

### 5.2 Design Pattern MVC

Per il sistema legacy abbiamo scelto di non modificare l'implementazione in design pattern MVC, illustrato di seguito.

## 5. FASE DI PROGETTAZIONE

---

Il design pattern **Model-View-Controller** (MVC) assegna agli oggetti in un'applicazione uno dei seguenti ruoli:

- Model
- View
- Controller

Il pattern non solo definisce i ruoli di ogni oggetto nell'applicazione, ma definisce anche il modo in cui gli oggetti comunicano tra di loro. Ognuno dei tre tipi di oggetti è separato dagli altri da confini astratti e comunica con gli altri attraverso questi confini. L'insieme di oggetti di un certo tipo MVC in un'applicazione è definito come "*layer*": nei paragrafi successivi ci riferiremo quindi agli insiemi di oggetti in questo modo.

I benefici di adottare questo pattern sono numerosi; molti oggetti in queste applicazioni tendono ad essere maggiormente riusabili, con interfacce meglio definite.

Le applicazioni che utilizzano il design MVC tendono ad essere più estendibili delle altre.

### 5.2.1 Oggetti Model

Gli oggetti Model racchiudono i dati di un'applicazione e definiscono la logica che manipola e processa quei dati. Ad esempio, nella piattaforma legacy, un oggetto Model rappresenta uno degli utenti o uno dei casi di plagio presenti. Un oggetto Model può avere relazioni uno-a-uno e uno-a-molti con altri oggetti model, quindi a volte il *Model layer* di un'applicazione effettivamente è visualizzabile come uno o più grafi di oggetti. Nella nostra applicazione gli utenti ed i casi di plagio sono in relazione uno-a-molti, in quanto ogni utente può inserire zero, uno o più casi di plagio.

I dati che fanno parte dello stato persistente di un'applicazione, sia essa in files che in database, dovrebbe risiedere in oggetti Model una volta caricati nell'applicazione.

## 5. FASE DI PROGETTAZIONE

---

Visto che gli oggetti Model rappresentano conoscenze relative al dominio di un problema, possono essere riutilizzati in altri domini simili.

Idealmente, un oggetto Model non dovrebbe avere alcuna connessione esplicita con gli oggetti View che presentano i suoi dati, in modo da non soffrire di problemi di interfaccia o presentazione.

Le azioni degli utenti nel *View layer* che creano o modificano dati sono comunicate attraverso un oggetto Controller e risultano nella creazione o aggiornamento di un oggetto Model. Quando un oggetto Model cambia ( ad esempio dopo un aggiornamento ), esso notifica un oggetto Controller, che aggiorna gli oggetti View appropriati.

### 5.2.2 Oggetti View

Un oggetto View è un oggetto in un'applicazione che gli utenti possono vedere.

Esso deve sapere come mostrarsi e rispondere alle azioni degli utenti.

La principale occupazione di un oggetto View è di mostrare dati dagli oggetti Model dell'applicazione e di permettere la modifica di quei dati. A dispetto di questo, in un'applicazione MVC gli oggetti View sono generalmente disaccoppiati dagli oggetti Model.

Nella nostra applicazione gli oggetti View sono rappresentati da layout XML che mostrano dati in maniera visivamente chiara.

Gli oggetti View si rendono conto dei cambiamenti nei dati degli oggetti Model attraverso gli oggetti Controller e comunicano le azioni degli utenti. Ad esempio, nella nostra applicazione, l'utente inserisce le informazioni inerenti un caso di plagio attraverso la View ed una volta inviata il Controller controlla e gestisce i dati, trasformandoli in oggetti Model.

### 5.2.3 Oggetti Controller

Un oggetto Controller fa da intermediario tra una o più oggetti View della piattaforma ed uno o più dei suoi oggetti Model. Gli oggetti Controller

## 5. FASE DI PROGETTAZIONE

---

sono quindi dei “canali” attraverso i quali gli oggetti View si rendono conto dei cambiamenti degli oggetti Model e viceversa.

Gli oggetti Controller possono anche preparare e coordinare tasks e gestire il ciclo di vita degli altri oggetti. Nella piattaforma legacy gli oggetti.

Controller sono astratti da metodi che sfruttano dei decoratori *@route*.

Un oggetto Controller interpreta le azioni degli utenti effettuate negli oggetti View e comunica il passaggio di nuovi dati o di aggiornamenti al *Model layer*.

Quando degli oggetti Model cambiano, un oggetto Controller comunica i dati del nuovo Model agli oggetti View, così da mostrare correttamente i dati aggiornati. Nella piattaforma legacy le *routes* si occupano di recepire i dati inviati dall’applicazione Android, per il loro inserimento, modifica e aggiornamento, e di reinviare i dati aggiornati mediante risposta, in modo da aggiornare la View.

Successivamente all’interno della View, i dati verranno controllati, inseriti e visualizzati correttamente nei layout.

### 5.3 Architettura del Sistema

Nello sviluppo dell’applicazione abbiamo mantenuto l’architettura Client- Server del sistema legacy, aggiungendo però un nuovo tipo di Client.

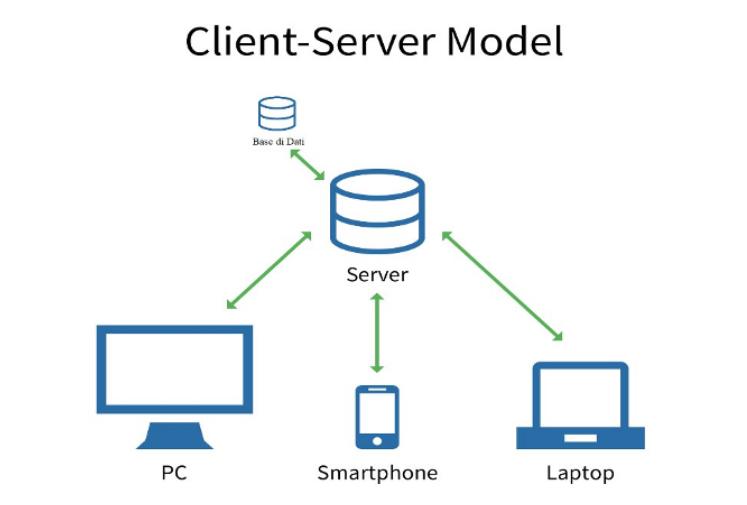
#### 5.3.1 Sistema Client-Server

In informatica un “*sistema client-server*” indica un’architettura di rete nella quale genericamente un computer client o terminale si connette ad un server per la fruizione di un certo servizio, quale ad esempio la condivisione di una certa risorsa hardware/software con altri client, appoggiandosi alla sottostante architettura protocollare.

## 5. FASE DI PROGETTAZIONE

---

Figura 5.1: Modello Client Server



La piattaforma legacy è stata progettata utilizzando una suddivisione in sottosistemi, permettendo uno sviluppo indipendente per ognuno dei sottosistemi.

### 5.3.2 SubSystem Main

Questo sottosistema si occupa delle funzionalità “principali” della piattaforma, ovvero tutto il processo di rilevazione del plagio.

Il calcolo dei valori di plagio avverrà sfruttando le funzionalità implementate sulla base del sistema legacy e il salvataggio dei dati avverrà in forma temporanea per il confronto corrente.

#### Funzionalità:

- Caricamento brani
- Calcolo dei valori del confronto
- Salvataggio dei dati del confronto corrente

### 5.3.3 SubSystem Sentences

Questo sottosistema si occupa delle funzionalità di gestione dei casi di plagio, ovvero la visualizzazione dei casi di plagio presenti sulla piattaforma,

## 5. FASE DI PROGETTAZIONE

---

la visualizzazione dei dati del singolo caso, la possibilità di inserimento di un caso di plagio, e l'aggiornamento dei dataset per l'*Ensemble* ed il *Clustering*; quest'ultima parte per permettere al sistema di rilevazione del plagio di affinare il proprio calcolo in base ai dati presenti sulla piattaforma.

**Funzionalità:**

- Richiesta tutti casi di plagio
- Richiesta dati del singolo caso di plagio
- Richiesta di ricerca di casi plagio
- Inserimento caso di plagio
- Aggiornamento dataset per *Ensemble* e *Clustering*

### 5.3.4 SubSystem Users

Questo sottosistema si occupa delle funzionalità di gestione degli account, ovvero la sua autenticazione nel sistema ( che permetterà l'accesso alle funzionalità di inserimento dati ), l'aggiornamento dei dati dell'account, e la possibilità di richiedere il reset della password ricevendo una email contenente un link per l'aggiornamento.

**Funzionalità:**

- Autenticazione
- Aggiornamento informazione dell'Account
- Reset PW

## 5.4 Tipologie di utenti

Al momento l'applicazione considera sue tipologie di utenti:

- **Utente Guest**, non registrato sulla piattaforma, il quale ha accesso alle funzionalità di plagio e può ricercare casi di plagio e visualizzarne i dettagli; non ha accesso all'inserimento di nuovi casi.

## **5. FASE DI PROGETTAZIONE**

---

- **Utente Tecnico**, registrato sulla piattaforma, il quale ha accesso a tutte le funzionalità del sistema, dalla rilevazione del plagio, ricerca e visualizzazione. Può anche inserire un nuovo caso, senza visionare i dati e senza seguire i passi della rilevazione convenzionale del plagio, questo permette di inserire velocemente casi di plagio già conosciuti allo scopo di popolare la piattaforma e raffinare quindi il calcolo.

---

---

# CAPITOLO 6

---

## FASE DI SVILUPPO

### 6.1 Tecnologie di implementazione

Di seguito una descrizione delle tecnologie utilizzate nell’implementazione delle funzionalità identificate nei capitoli precedenti.

#### 6.1.1 Activity e Fragment

Un’activity in Android è essenzialmente una finestra che contiene l’interfaccia utente di un’applicazione ed il suo scopo è quello di permettere un’interazione con gli utenti. Un’applicazione Android può avere zero o più activity, anche se solitamente almeno una è presente.

Dal momento in cui un’activity compare sullo schermo al momento in cui scompare essa passa attraverso una serie di stati, che, nel loro complesso, rappresentano il ciclo di vita dell’activity (life cycle). Comprendere il ciclo di vita di un’activity è fondamentale per assicurarci che le nostre applicazioni Android funzionino correttamente.

All’interno delle Activity possono essere presenti uno o più Fragment, ovvero delle sottoclassi che permettono di modularizzare l’Activity principale. In questo

modo sarà possibile mostrare, nascondere, modulare o posizionare i diversi Fragment in base alla dimensione della device su cui eseguiamo l'applicazione.

### 6.1.2 Dataset, Ensemble e Clustering

Le funzionalità di *Ensemble* e *Clustering* sono basate, come detto, sul sistema legacy. Tale sistema vanta una “meta-euristica adattiva”, adattiva in quanto sfruttando dei dataset, per l'*Ensemble* ed il *Clustering*, affina ogni volta il suo calcolo grazie ai nuovi dati.

L'*Ensemble*, suddiviso in due parti, contiene diverse coppie di brani, sia in plagio che non in plagio; mentre il *Clustering* contiene i singoli brani caricati sulla piattaforma per poterli suddividere in “gruppi” simili per le funzionalità di *Clustering*; successivamente ci riferiremo a questi set di dati per raffinare il calcolo del plagio come dataset o dataset *Ensemble* o dataset *Clustering*.

### 6.1.3 Flask Blueprint

Sul server legacy sono utilizzati diversi contenitori per le diverse funzionalità, suddivisi in base al loro ambito; con il fine di suddividere logicamente il progetto in sottosistemi. Più precisamente si sono utilizzate le Flask Blueprint.

In questo modo è stato possibile, agendo sui singoli moduli, applicare le modifiche necessarie alla comunicazione con la nostra applicazione Android senza dover stravolgere la struttura originaria.

Le Blueprint di Flask encapsulano le **funzionalità** del sistema.

Ogni Flask Blueprint è un **oggetto** che funziona in maniera molto simile ad una applicazione Flask.

Blueprint Flask e Applicazioni Flask, infatti, possono entrambi avere risorse, come file statici, templates, e Views associate alle *routes*. Ad ogni modo una Flask Blueprint non è una applicazione; deve essere registrata in un'applicazione prima di poterla far partire. Quando si registra una Blueprint in una applicazione, si sta effettivamente **estendendo** l'applicazione con i

contenuti della Blueprint.

**Le Blueprint registrano operazioni che potranno essere eseguite una volta registrate su di una applicazione.**

### 6.1.4 Models

I Models del pattern MVC sono rappresentati come classi all'interno della piattaforma legacy, e derivano da db.Model, sfruttando le funzionalità di SQLAlchemy [3.4.1]; le loro istanze potranno essere quindi trattate come oggetti nel database. Inoltre è proprio da queste classi che vengono create le tabelle nel database, utilizzando il metodo `db.create_all()` da console. Gli attributi di queste classi sono definiti tramite i metodi `db.column( settings )`, derivanti sempre da SQLAlchemy, dove per `settings` intendiamo l'inserimento della tipologia di attributo, se esso è chiave primaria ed altri vincoli per generare la tabella nel database.

Nel sistema legacy erano presenti due Models, `User` e `Sentence`, su cui è stato possibile lavorare per aggiornare le funzioni:

- **User** è una classe derivante sia da db.Model che da UserMixin; quest'ultimo, importato da Flask-Login [3.4.2], segnala alle sue funzionalità che la classe rappresenta l'utente, offrendo diverse funzionalità standard per esso. Alla classe `User` sono stati successivamente aggiunti dei metodi per la generazione e la verifica di un token per la gestione del reset della password.

In aggiunta ai suoi attributi comuni, `User` ha l'attributo `sentences` che rappresenta una **relazione** con il Model `Sentence`.

- **Sentence** è una classe derivante soltanto da db.Model, ed anch'essa è definita grazie alle funzionalità di sqlalchemy, allo stesso modo di `User`. In aggiunta ai suoi attributi comuni, `Sentence` ha l'attributo `user_id` che rappresenta una **relazione** con il Model `User`. La relazione descritta rappresenta una relazione uno a molti, dove ogni `User` è considerato autore di più `Sentence`, ed ogni `Sentence` ha un unico autore.

### 6.1.5 Routes

Come accennato nel design pattern MVC, la parte Controller della piattaforma legacy è gestita da metodi decorati con `@route`.

Questo decoratore permette a Flask di collegare uno specifico URL alle funzionalità di quel metodo, richiamando il metodo qualora l'utente acceda alla pagina. Ad esempio il metodo `loginMobile()` è decorato con `@user.route('/loginMobile')`, dove `User` rappresenta la blueprint corrente, ed `'/loginMobile'` l'URL collegato alla funzione.

Tramite richieste GET e POST è poi possibile interagire con le singole route ed ottenere risposte dal server. Questo è il funzionamento basilare di una `route`, utilizzato per tutte le funzioni Controller della nostra applicazione, successivamente ci riferiremo a questi metodi semplicemente come '`routes`'; eccezioni e particolarità saranno evidenziate nella descrizione delle singole funzionalità.

### 6.1.6 Icône utilizzate nei casi di plagio

Il sistema utilizza delle coppie di icône associate ai casi di plagio per comunicare visivamente all'utente se il caso è considerato un plagio o meno secondo il sistema, per comunicare lo stato giuridico del caso di plagio e per segnalare la presenza di un verdetto in formato PDF.

-  Secondo il sistema, il caso con questa icona è **considerato Plagio**.
-  Secondo il sistema, il caso con questa icona **non è considerato Plagio**.
-  Il caso con questa icona **non fa parte di un processo in tribunale**.
-  Il caso con questa icona **fa parte di un processo in tribunale, esso non è concluso e non esiste quindi un verdetto**.

-  Il caso con questa icona **fa parte di un processo in tribunale, esso è concluso e esiste quindi un verdetto.**
-  Il caso con questa icona **permette il download** del verdetto associato.

## 6.2 IDE Utilizzate

### 6.2.1 PyCharm

Per le modifiche alla piattaforma legacy abbiamo utilizzato Pycharm Community Edition. PyCharm è una “integrated development environment” (IDE) utilizzata per la programmazione e specializzata in linguaggio Python, sviluppata dalla compagnia ceca JetBrains [19] e fornisce:

- Code analysis
- Graphin Debugger
- Unit Tester integrati
- Integrazione con i sistemi di version control (VCsEs).
- Supporto allo sviluppo web con Django,
- Supporto al data science con Anaconda.

PyCharm è cross-platform, con versioni per Windows, macOS e Linux, ed è disponibile sia nel formato “Community Edition”, rilasciata sotto la licenza Apache, sia nel formato “Professional Edition”, rilasciata sotto licenza proprietaria che include funzionalità Enterprise.

### 6.2.2 Android Studio

Per lo sviluppo dell'applicazione Android abbiamo utilizzato Android Studio.

Android Studio è utilizzato per la programmazione e il design di applicazioni Android sia in linguaggio Java che in linguaggio Kotlin sviluppato direttamente da Google.

Fornisce innumerevoli funzionalità, tra cui notiamo principalmente:

- Code analysis
- Graphic Debugger
- Emulatore completo
- Compatibilità con tutte le versioni di Android rilasciate

Android Studio è parte dell'Android Open Source Project, e per questo è gratuito.

### 6.3 Fasi di implementazione

#### 6.3.1 Definizione della struttura del progetto

Il primo aspetto analizzato e sviluppato è stato quello della suddivisione del progetto in activity e fragment, in accordo con la scomposizione modulare definita in fase di progettazione. Successivamente è stata definita una divisione di compiti per quanto riguarda lo sviluppo del Back-End (riguardante la connessione con il server legacy, l'adattamento delle funzioni sullo stesso affinchè comunichino correttamente con l'applicazione, e la stesura delle classi Java dell'applicazione Android) affidato al sottoscritto, e che sarà oggetto principale di discussione di questa tesi, e il Front-End affidato al mio collega Raffaele Squillante [21].

#### 6.3.2 Permessi Android

Qualunque applicazione installata su un dispositivo, necessita di chiedere all'utente l'autorizzazione da usare servizi, sensori o funzionalità del sistema esterni all'applicazione stessa.

A tal proposito, agli utenti che fruiranno della nostra applicazione sono richiesti solamente tre permessi:

- **Internet**: per le richieste al Server
- **Read External Memory**: per leggere i file presenti sulla memoria e poterli inviare al Server
- **Write External Memory**: per scaricare sul File System i verdetti richiesti

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.plagiomusicale">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Figura 6.1: Snippet del Manifest della nostra applicazione Android dedicato ai permessi

### 6.3.3 Inizializzazione e configurazione del progetto

All'interno del progetto generato da Android Studio, sono state create le Activity e i Fragment necessari.

Fatto ciò, è stata impostata l'Activity *activity\_login*, la quale permette agli utenti di autenticarsi o procedere come ospiti, come attività di avvio della piattaforma.

### 6.3.4 Gestione della connessione al server legacy

Dal momento che il server legacy è basato su di una piattaforma Flask che opera principalmente con dei *Form* scambiati tra Client e Server (su di cui vengono effettuate le operazioni), ed inoltre il Server si occupava anche di restituire al Client la visualizzazione dei risultati (sotto forma di Form o di parti di pagina vere e proprie), si è vista necessaria l'attuazione di alcune modifiche.

Tutte le funzioni, che analizzeremo in seguito, sono state ricreate e modificate

per gestire richieste **GET** e **POST**, e per ricevere gli input necessari non più tramite *Form* ma tramite parametri GET o POST.

Una volta ricevuti correttamente i parametri dal Client, le funzioni operano su questi ultimi così *come in precedenza* operavano con i campi del Form. **Da questo punto di vista il codice viene totalmente riutilizzato e non vi è nessuna modifica a livelli più bassi.**

L'output delle funzioni, *e di conseguenza l'invio al Client*, viene gestito, in base alle funzioni, con dei file **JSON** (per riprendere la struttura dei Form o per inviare i risultati delle query sulla base di dati), o con delle *semplici stringhe* per esiti.

### 6.3.5 Sviluppo delle funzionalità

La prima funzionalità sviluppata è stata la **rilevazione del plagio**, essendo il nucleo dell'attività di tesi, mediante Fragment e modificando la blueprint *Main* del sistema legacy. Il passo successivo è stato poi procedere allo sviluppo delle **funzionalità dedicate agli utenti** con il Fragment UserPage e lavorando sulla blueprint *Users* e relativo Model del sistema legacy. Successivamente è stata adattata la blueprint *Sentences*, dipendente anche da *Main*, la quale permette **l'inserimento e la gestione dei casi di plagio** e delle loro informazioni all'interno della nostra piattaforma. Sono stati quindi sviluppati:

- una Activity che permette l'autenticazione degli utenti
- una Activity che contiene il menù, e lo spazio per la visualizzazione dei vari Fragment
- un Fragment per la visualizzazione e la ricerca dei casi presenti sulla piattaforma;
- un Fragment per la visualizzazione dei dettagli sul singolo caso;
- un Fragment per il confronto di due brani;

- un Fragment per l'inserimento di un nuovo caso sulla piattaforma
- un Fragment per l'area utente.

### 6.4 Menu

In base al tipo di utente [Paragrafo 5.4] è stato creato un differente menù, che viene poi caricato nell'Activity Menu.

In questo modo sarà impossibile ad utenti guest poter accedere a funzioni a loro non dedicate.

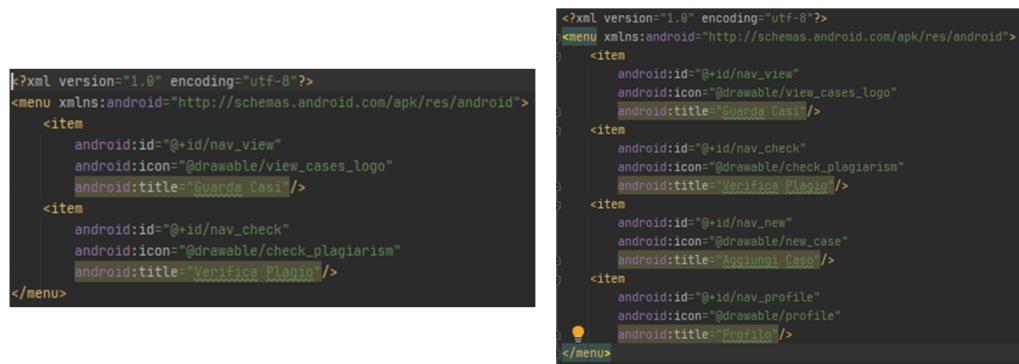


Figura 6.2: Menu per guest, e menu per utente tecnico

### 6.5 SubSystem Main

#### 6.5.1 Caricamento brani

La funzionalità di caricamento dei brani è stata implementata tramite un Fragment, contenente i campi rappresentanti i brani, ai quali sono stati imposti dei vincoli sui formati caricabili, restringendo il caricamento dei files al solo formato xml.

Sul sistema legacy è presente, inoltre, un ulteriore metodo di validazione per

controllare che l’utente non possa caricare lo stesso file in entrambi i campi. Una volta inviati i dati al server, tramite richiesta POST, e una volta validati quest’ultimi, i brani vengono salvati nelle cartelle legacy e nel sistema come parte dei dati del confronto corrente.

```
if request.method == 'POST':
    song1 = request.files['song1']
    song2 = request.files['song2']
    save_songs_in_legacy_and_static(song1, song2)
```

Figura 6.3: Ricevuti i file da richiesta POST, il server li valida e poi li salva (tramite funzione legacy)

### 6.5.2 Calcolo dei valori del confronto

La funzionalità di calcolo dei valori del confronto tra brani, avendo già ricevuto i campi necessari dal Client, riutilizza totalmente funzionalità estrapolate dal progetto legacy.

Essa si occupa di:

- Convertire i brani da file xml a rappresentazioni testuali.
- Calcolare i risultati dell’*Ensemble* e/o del *Clustering* sulle suddette rappresentazioni.
- Valutare e suddividere i valori calcolati in un oggetto Dictionary (oggetto Python).

```
dictionary, labels, values, val_clustering, val_threshold, avg, result_of_confrontation, file_name1, file_name2\= calculate_results_and_informations()
```

Figura 6.4: Funzione legacy che calcola i risultati del confronto tra i brani appena caricati

### 6.5.3 Salvataggio dei dati del confronto corrente

Il salvataggio dei dati dopo un confronto atto alla rilevazione del plagio viene eseguito tramite dei metodi che si occupano della formattazione e del salvataggio delle informazioni del confronto corrente in maniera incrementale durante il calcolo, anche loro sono rimaste invariate dal progetto legacy.

```
write_result_values_in_static_temp(dictionary, avg, result_of_confrontation, file_name1, file_name2)
lines = [] # empty list
file_path = os.path.join(current_app.root_path, 'static/last_check_temp_files/result_values.txt')
```

Figura 6.5: Funzione legacy che salva i dati del confronto corrente

### 6.5.4 Visualizzazione e schematizzazione dei risultati

La visualizzazione dei risultati è gestita dal Fragment *Result* che riceve dal server un file JSON contenente tutti i dati relativi al caso in esame.

All'interno del Fragment sono presenti diversi grafici, modellati in base ai dati ricevuti.

Le altre informazioni necessarie vengono visualizzate in diversi segmenti del Fragment mediante delle TextView.

## 6. FASE DI SVILUPPO

---

```
try:
    with open(file_path) as data_file: # apro il file result_values.txt in lettura
        for row in data_file: # per ogni riga
            lines.append(row.strip('\n')) # salvo nella nostra lista e tolgo i newline
except OSError as error: # errore
    print(error)
lines = ['0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0'] # finti risultati
print(lines)
print("post")
dataResult = {
    "file_not_found": False,
    "cosine": lines[3],
    "sorensen_dice": lines[4],
    "overlap": lines[5],
    "clustering": lines[6],
    "clusteringValue": lines[7],
    "percentage": lines[8],
    "threshold": lines[9]
}
if lines[7] == '0':
    plagiarism = False
else:
    plagiarism = True
return jsonify({"plagiarism": plagiarism, "dataResult": dataResult}) #Ritorno JSON Risultati
```

Figura 6.6: Estrapolazione dal modello dati legacy dei dati necessari e creazione del file JSON di output

```
if (result.isPlagiarism())
    plagiarismTW.setText("DETECTED");
else
    plagiarismTW.setText("NOT DETECTED");
String[] cosine = result.getDataResult().getCosine().replace( oldChar: '.', newChar: ',' ).split( regex: "[,]" );
System.out.println(cosine[0]);
int cosineInt = Integer.parseInt(cosine[0]);
String[] overlap = result.getDataResult().getOverlap().replace( oldChar: '.', newChar: ',' ).split( regex: "[,]" );
int overlapInt = Integer.parseInt(overlap[0]);
String[] sorensen = result.getDataResult().getSorensen_dice().replace( oldChar: '.', newChar: ',' ).split( regex: "[,]" );
int sorensenInt = Integer.parseInt(sorensen[0]);
cosineProgress.setProgress(cosineInt);
cosineTW.setText(result.getDataResult().getCosine());
sorensenProgress.setProgress(sorensenInt);
sorensenTW.setText(result.getDataResult().getSorensen_dice());
overlapProgress.setProgress(overlapInt);
overlapTW.setText(result.getDataResult().getOverlap());
percentualeTW.setText(result.getDataResult().getPercentage());
thresholdTW.setText(result.getDataResult().getThreshold());
clusteringTW.setText(result.getDataResult().getClustering());
return view;
```

Figura 6.7: Modellazione della View del Fragment in base ai dati ricevuti (contenuti in result)

## 6. FASE DI SVILUPPO

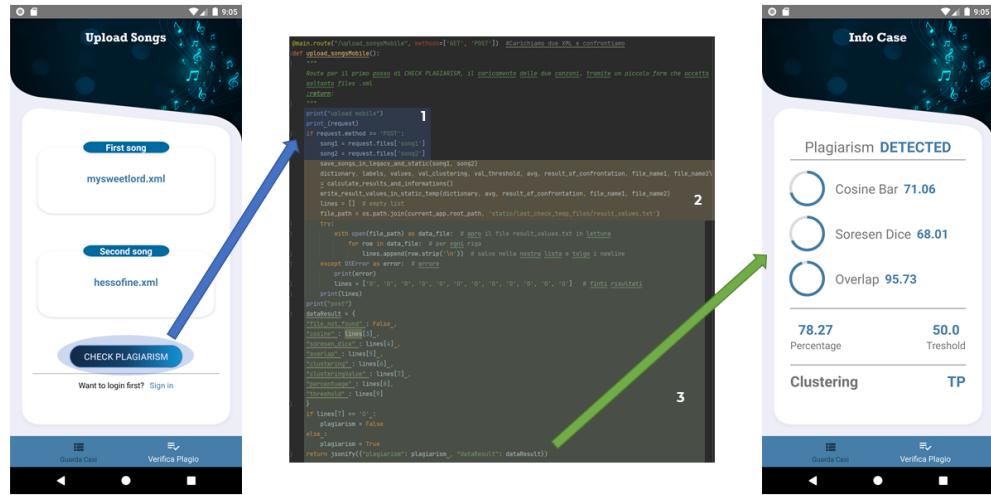


Figura 6.8: Tutti i passi del caricamento di due brani per il confronto del plagio lato server

## 6.6 SubSystem Users

### 6.6.1 Autenticazione

La funzionalità di autenticazione è implementata con una Activity che permette l’accesso utilizzando email e password. La *route* di login si occupa dei controlli di login seguenti.

Si controlla che l’utente sia presente nel database tramite l’email inserita, dopodiché si effettua l’hashing ed il controllo della password sfruttando le funzionalità di **Flask-Bcrypt** [vedi Flask-Bcrypt 3.4.3]. Nel caso l’utente si autentichi, l’Activity reindirizza l’utente alla Activity *Menu* con il Fragment di **visualizzazione dei casi di plagio**.

## 6. FASE DI SVILUPPO

---

```
@users.route("/loginMobile", methods=['GET', 'POST'])
def loginMobile():
    """
    Route per il login di un utente, permette di accedere al form e di effettuare il login in caso
    di submit del suddetto form
    :return:
    """

    mail = request.args.get('mail', None)
    password = request.args.get('pw', None)
    user = User.query.filter_by(email=mail).first()
    if user and bcrypt.check_password_hash(user.password, password): # if user exists and pass matches
        return "1" # LOGIN OK -> REDIRECT AL MENU
    else:
        return "0" # NO LOGIN -> ERRORE
```

Figura 6.9: Funzione del server che, ricevuti i dati dal Client, processa l'autenticazione

```
@Override
protected void onPostExecute(String string) {
    super.onPostExecute(string);
    while(!trovato){} //Aspettiamo la risposta dal server
    Intent login = new Intent(context, Menu.class);
    login.putExtra(name: "mail",string); //Salviamo la mail dell'utente autenticato
    context.startActivity(login); //Reindirizziamo al Menu
    System.out.println("LOGIN COME: "+string);
}
```

Figura 6.10: Funzione del client che, in caso di autenticazione corretta, reindirizza l'utente al menu (salvando la mail in sessione)

## 6. FASE DI SVILUPPO



Figura 6.11: Tutti i passi del login di un utente tecnico

### 6.6.2 Visualizzazione e Aggiornamento informazione dell'account

La funzionalità di visualizzazione e aggiornamento account è stata implementata dal Fragment UserArea che mostra all'utente, mediante delle EditText, le informazioni del proprio profilo. Qui l'utente può cambiare o lasciare inalterati alcuni valori.

Al click sul bottone *Update* viene inviata una richiesta al server, il quale una volta processati i dati, aggiorna i valori nel database.

Dopodiché viene ricaricato il Layout del Fragment, dove si visioneranno i dati aggiornati.

## 6. FASE DI SVILUPPO

---

```
@Override
protected Integer doInBackground(String... strings) {
    OkHttpClient client = new OkHttpClient();
    String url = "http://10.0.2.2:5000/accountMobile?mail="+strings[0];
    System.out.println(url);
    System.out.println(strings.length+" LUNGHEZZA PARAMETRI");
    Request request = new Request.Builder().url(url).build();
    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(@NotNull Call call, @NotNull IOException e) {
            e.printStackTrace();
        }

        @Override
        public void onResponse(@NotNull Call call, @NotNull Response response) throws IOException {
            String jsonStr = response.body().string();
            ObjectMapper mapper = new ObjectMapper();
            userData = mapper.readValue(jsonStr,UserData.class);
            System.out.println("DOWNLOAD DI "+userData);
            terminatoUser=true;
        }
    });
    while(!terminatoUser){
        System.out.println("Non ho ancora finito");
    }
    return 0;
}
```

Figura 6.12: Richiesta Client per l'utente corrente

```
@users.route("/accountMobile", methods=['GET', 'POST'])
def accountMobile():
    """
    Route per visualizzare i dati di un account
    :return:
    """

    mail = request.args.get('mail', None)
    user = User.query.filter_by(email=mail).first()
    return createJsonUser(user)
```

Figura 6.13: Gestione della richiesta dati utente sul server

## 6. FASE DI SVILUPPO

---

```
def createJsonUser(user):
    id = user.id
    username = user.username
    first_name = user.first_name
    last_name = user.last_name
    email = user.email
    image_file = user.image_file
    password = user.password
    expertise = user.expertise

    userData = {
        "id": id,
        "username": username,
        "first_name": first_name,
        "last_name": last_name,
        "email": email,
        "image_file": image_file,
        "password": password,
        "expertise": expertise
    }

    return jsonify(userData)
```

Figura 6.14: Creazione JSON con Dati Utente

```
@users.route("/accountUpdateMobile", methods=['GET', 'POST'])
def accountUpdateMobile():
    """
    Route per visualizzare i dati di un account
    :return:
    """

    if request.method == 'POST':
        mail = request.form.get("mail")
        user = User.query.filter_by(email=mail).first() #Cerco utente corrispondente
        if request.form.get("username") is not None: #Se ha inserito un username
            username = request.form.get("username")
            user.username = username #Aggiorno Username
        if request.form.get("expertise") is not None: #Se ha inserito delle expertise
            expertise = request.form.get("expertise")
            user.expertise = expertise #Aggiorno Expertise
        db.session.commit() #Aggiorno Utente in DB
        return "OK"
    return "NO"
```

Figura 6.15: Gestione della richiesta aggiornamento dati utente sul server

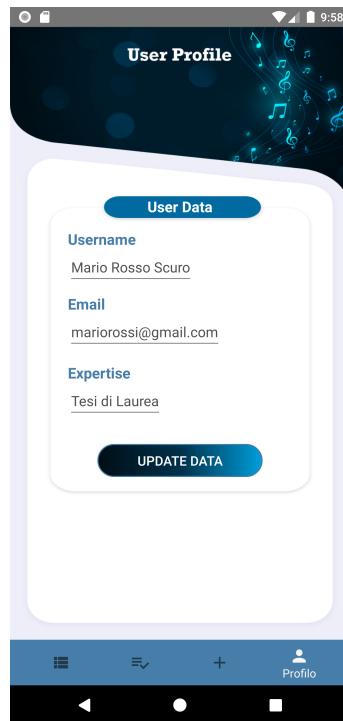


Figura 6.16: Fragment UserArea

## 6.7 SubSystem Sentences

### 6.7.1 Visualizzazione casi di plagio

La funzionalità di visualizzazione dei casi di plagio presente sull'applicazione è implementata mediante il Fragment View Cases che, utilizzando una richiesta GET al server, riesce ad ottenere tutti i Model Sentence presenti.

Successivamente si occupa di popolare il List View con tutti i casi scaricati e di visualizzarli tramite un custom adapter.

## 6. FASE DI SVILUPPO

---

```
@Override
protected Integer doInBackground(String... strings) {
    OkHttpClient client = new OkHttpClient();
    String url;
    if(strings.length==0) //Se non ci sono filtri di ricerca
        url = "http://10.0.2.2:5000/all_sentencesMobile"; //Tutti i Casi
    else //Se ci sono filtri di ricerca
        url = "http://10.0.2.2:5000/search_pageMobile?title="+strings[1]+"&author="+strings[0]+"&info="+strings[2]; //Ricerca
    Request request = new Request.Builder().url(url).build(); //Richiesta al Server
    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(@NotNull Call call, @NotNull IOException e) {
            e.printStackTrace();
        }

        @Override
        public void onResponse(@NotNull Call call, @NotNull Response response) throws IOException {
            String jsonStr = response.body().string(); //Scarico il JSON
            ArrayList<HashMap<String, String>> caseList = new ArrayList<>();
            ObjectMapper mapper = new ObjectMapper();
            sentenzeArray = mapper.readValue(jsonStr,Case[].class); //Mappo il file JSON in un Array di Casi
            for(int i=0;i<sentenzeArray.length;i++)
            {
                caseList.add(sentenzeArray[i]);
            }
            terminato=true; //Comunico la fine del Download
        }
    });
    return 0;
}
```

Figura 6.17: Richiesta Client dei casi sul server

```
@sentences.route('/all_sentencesMobile')
def all_sentencesMobile():
    """
    Route per la visualizzazione di tutte le sentence contenute nel database (paginate)
    :return:
    """

    page = request.args.get('page', 1, type=int) # prendiamo il parametro page, se non c'è è 1, tipo int
    sentences = Sentence.query.order_by(Sentence.date_posted.desc())
    print("all_sentencesMobile")
    return createJson(sentences)
```

Figura 6.18: Gestione della richiesta dei casi sul server

## 6. FASE DI SVILUPPO

---

```
def createJson(sentences):
    arraySentences = []
    count = 0
    for s in sentences:
        dataR = read_result_values_file_from_sentence_folder(s.id)
        if dataR[0] == '0' and dataR[0] == '0' and dataR[0] == '0':
            file_not_found = True
        else:
            file_not_found = False
        dataResult = {
            "file_not_found": file_not_found,
            "cosine": dataR[3],
            "sørensen_dice": dataR[4],
            "overlap": dataR[5],
            "clustering": dataR[6],
            "clusteringvalue": dataR[7],
            "percentage": dataR[8],
            "threshold": dataR[9]
        }
        if s.has_trial == s.has_verdict == True:
            pathFile = "/static/saved_sentences/" + str(s.id) + "/verdict.pdf"
        else:
            pathFile = "empty"
        arraySentences.append({
            "id": s.id,
            "title": s.title,
            "first_song": s.first_song,
            "second_song": s.second_song,
            "date_posted": s.date_posted,
            "is_plagiarism": s.is_plagiarism,
            "has_trial": s.has_trial,
            "has_verdict": s.has_verdict,
            "pathfile": pathFile,
            "info": s.info,
            "user_id": s.user_id,
            "dataResult": dataResult
        })
        count += 1
    return jsonify(arraySentences)
```

Figura 6.19: Creazione del file JSON con i risultati

```
private void updateGUI(ArrayList<Case> sentenze)
{
    customAdapter.clear();
    for(int i=0;i<sentenze.size();i++)
    {
        customAdapter.add(sentenze.get(i));
        customAdapter.notifyDataSetChanged();
    }
    customAdapter.notifyDataSetChanged();
    listView.setAdapter(customAdapter);
    terminato=false;
}
```

Figura 6.20: Aggiornamento dell’interfaccia del Client

## 6. FASE DI SVILUPPO

---

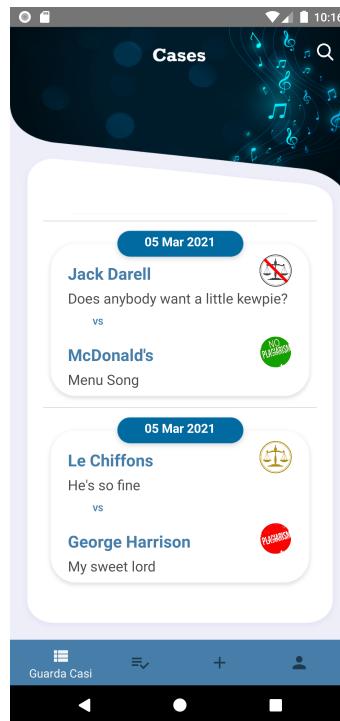


Figura 6.21: Fragment ViewCases

### 6.7.2 Visualizzazione singolo caso di plagio

La visualizzazione di un singolo caso di plagio è implementata da un Fragment Result che, prendendo come input il caso del List View su cui l'utente ha cliccato, popola i propri campi con i dati relativi al caso in esame.

Se il caso dovesse avere un verdetto, verrà caricato un Layout con la visualizzazione dell'immagine di quest'ultimo che permetterà di scaricarne il PDF. In caso contrario, non apparirà tale porzione di interfaccia.

L'utente ha la possibilità di visionare tutte le informazioni disponibili nel sistema sul caso selezionato. Le informazioni sono state suddivise nel seguente modo:

- Titolo del caso
- Titoli delle due canzoni
- Grafici dei risultati

## 6. FASE DI SVILUPPO

- Informazioni aggiuntive e visualizzazione del pdf del verdetto (in caso di assenza di quest'ultimo, l'icona non è presente)

```
@Nullable  
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
    Bundle bundle = getArguments();  
    View view;  
    selezionato= (Case) bundle.getSerializable( key: "selezionato");  
    if(selezionato.getHas_verdict()) //Se il caso ha un verdetto caricato, funzioni i layout dedicati a lui  
{  
        view = (View) inflater.inflate(R.layout.results_verdict, container, attachToRoot: false);  
        download=(Button) view.findViewById(R.id.download);  
        download.setOnClickListener(new View.OnClickListener(){  
            @Override  
            public void onClick(View v) {  
                String url = "http://10.0.2.2:5000"+selezionato.getPathFile(); //Path File del Verdetto sul Server  
                System.out.println(url);  
                DownloadManager.Request request = new DownloadManager.Request(Uri.parse(url))  
                    .setTitle("Verdetto "+selezionato.getTitle())  
                    .setDescription("Scaricando")  
                    .setNotificationVisibility(DownloadManager.Request.VISIBILITY_VISIBLE_NOTIFY_COMPLETED)  
                    .setRequiresCharging(false)  
                    .setAllowedOverMetered(true)  
                    .setAllowedOverRoaming(true); //Richiesta del File  
                System.out.println(request.toString());  
                DownloadManager downloadManager = (DownloadManager) getActivity().getSystemService(Context.DOWNLOAD_SERVICE);  
                downloadManager.enqueue(request); //Download del File dal Server  
            }  
        });  
    }  
}
```

Figura 6.22: Funzione per il Download del verdetto, dove presente

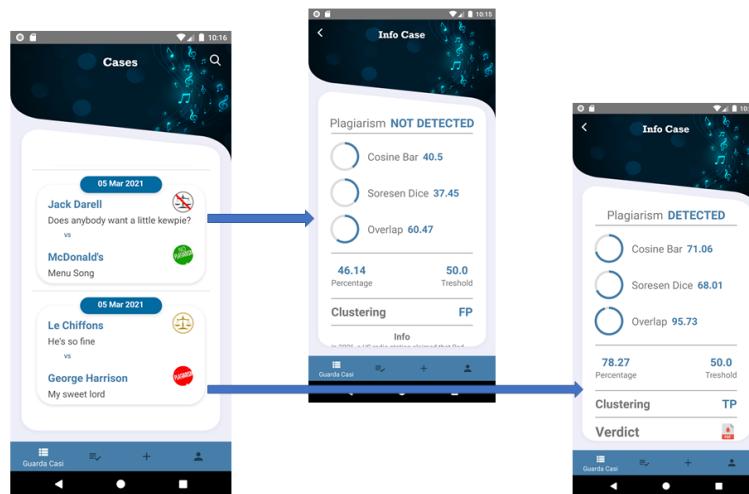


Figura 6.23: I Layout per i casi con verdetto, e quelli senza

### 6.7.3 Ricerca casi di plagio

La funzionalità di ricerca è implementata tramite una parte del Fragment View Cases, che permette l'inserimento di valori in tre campi: Il titolo di un brano, il nome di un artista e informazioni contenute nel campo info.

Al click del bottone l'applicazione invierà al server una richiesta GET con i dati inseriti dall'utente. L'applicazione permette l'inserimento completo o parziale dei titoli e delle informazioni aggiuntive. Il server, quindi, sfruttando i metodi di SQLAlchemy, carica dal database una lista di casi che rispettano i filtri inseriti e invia tramite file JSON i risultati all'applicazione, la quale aggiorna la List View di ViewCases visualizzando solamente i casi risultato della ricerca. [Figura 6.15]

### 6.7.4 Inserimento caso di plagio

L'inserimento di casi di plagio, funzionalità per utenti tecnici registrati, si può effettuare come inserimento diretto nel sistema utilizzando un form completo.

L'inserimento dei dati avviene tramite un *Fragment*, contenente i dettagli del caso di plagio.

Ad inserimento effettuato, e dopo aver cliccato sul bottone per procedere, verrà inviata una richiesta POST al server, il quale tramite la *route* del sistema legacy richiama le funzionalità di **aggiornamento dell' Ensemble e del Clustering** e procede all'inserimento del caso di plagio nel database.

I dati del confronto, salvati temporaneamente dal subsystem *Main* durante il calcolo, vengono ricopiatati in cartelle statiche associate all'istanza del database. L'utente viene poi reindirizzato al Fragment **della visualizzazione dei casi di plagio**.

## 6. FASE DI SVILUPPO

Figura 6.24: Invio della Richiesta dal Client con i dati del nuovo caso da inserire

```


```

sentences.route["/new_sentence_no_check/mobile", "method": "GET", "path": "/"], _def new_sentence_no_checkMobile():

Route per l'inscrizione di un poso di piovia. In questo caso si accede al form e una volta effettuato il submit, tutti i coloala verranno fatti in background.
Nota: Vi si accede da NEW CASE nella navabar
    return """
    """

if request.method == 'POST':
    #recupero i parametri della richiesta
    title = request.form.get("title")
    first_song_name = request.form.get("first_song_name")
    second_song_name = request.form.get("second_song_name")
    first_song = request.files[songi]
    second_song = request.files[sondo]
    info = request.form.get("info")
    mail=request.form.get("mail")
    radiochoice=request.form.get("radioChoice")
    radiochoice=request.form.get("radioChoice")
    user=User.query.filter_by(email=mail).first() #recupero l'utente che ha fatto la richiesta
    sentence = Sentence(firstSong=first_song_name, secondSong=second_song_name,
                        author=user.username, authorUser=user.id, TP=True #prova cont
    )

    if radiochoice == "haFratelli": #ha fratelli
        sentence.has_fratelli = True
        sentence.has_verdict = True
        verdict=request.files["verdict"]
    if radiochoice == "haFratelli":
        sentence.has_fratelli = True

```


```

Figura 6.25: Gestione della richiesta sul server per l'inserimento di un nuovo caso

## 6. FASE DI SVILUPPO

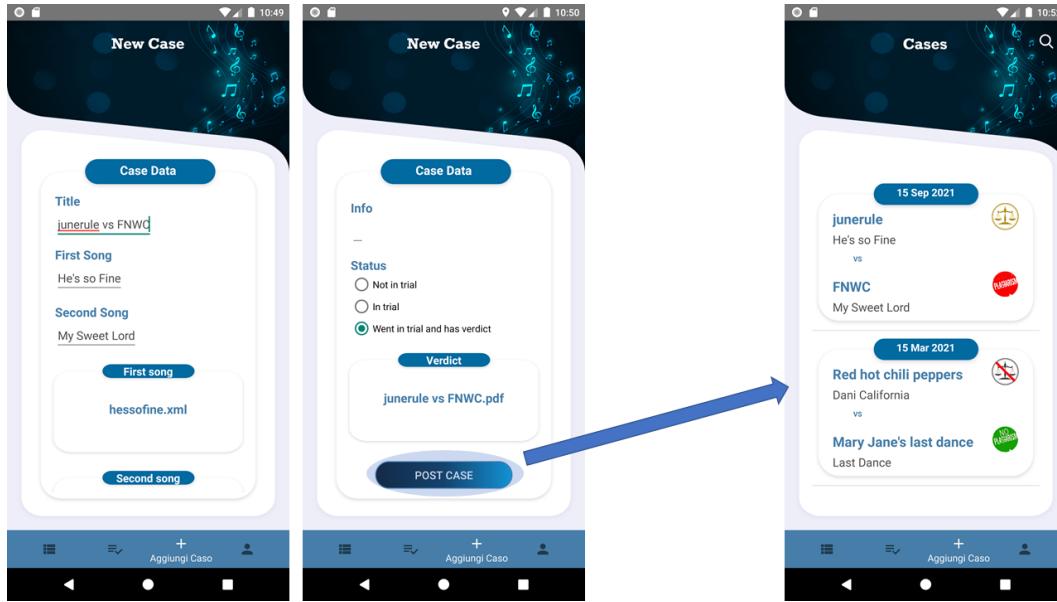


Figura 6.26: I passi per l'inserimento di un nuovo caso

### 6.7.5 Aggiornamento dataset Ensemble e Clustering

Le funzionalità di aggiornamento dei dataset per l'Ensemble ed il Clustering non sono state modificate, e quindi sono le stesse del sistema legacy.

Esse coinsistono nei metodi *update\_ensamble\_dataset()* ed *update\_clustering\_dataset()*.

## 6.8 Sviluppi futuri

In futuro si potrebbe migliorare l'applicazione e il server legacy con funzionalità aggiuntive o aggiornando funzionalità preesistenti. Ad esempio si è pensato a :

- Una funzione di ricerca più complessa, capace magari di riconoscere dei casi di plagio simili tra loro, per permettere ad un giudice di confrontarli, utilizzando il confronto come ulteriore supporto alle decisioni.
- Un aggiornamento o ampliamento della metaeuristica sulla quale si basa la rilevazione del plagio, inserendo magari la possibilità di aggiungere e

## **6. FASE DI SVILUPPO**

---

confrontare armonia e ritmo, o magari inserendo, oltre alla musica, anche il testo di una canzone. Questo per permettere un confronto, oltre che sulle parole utilizzate, anche sul significato del testo.

- L'utilizzo di thread in parallelo, per poter gestire più richieste in contemporanea.
- L'inserimento di un sistema di conferma alla registrazione da parte di un amministratore, per permettere soltanto ad utenti verificati come tecnici di poter accedere alle funzionalità di inserimento dati. Di conseguenza l'aggiunta di un nuovo utente amministratore.

---

---

## CAPITOLO 7

---

### RINGRAZIAMENTI

*Alla mia famiglia, a Mamma, a Papà, a Giuseppe e ad Elisabetta, che con i loro sforzi, di pazienza, di salute e di tasca, mi han sempre supportato e spronato a dare il meglio.*

*Ai nonni, amici silenziosi, con una parola, e non solo, di conforto sempre pronta al momento delle necessità.*

*A Victoria, compagna splendida che mi ha accompagnato durante tutto questo percorso, sempre pronta a riprendermi quando qualcosa poteva essere fatto meglio, ma sempre pronta a consolarmi nei momenti di sconforto.*

*Ad Emilio, a Francesca, a Davide, e a Jacopo, compagni fidati, che si sono sacrificati tutti, lavorando anche per me, per aiutarmi a coronare questo sogno nel migliore dei modi.*

*Ad Andrea, a Francesca, a Ignazio, a Marco, a Tammaro, amici, puri, sempre pronti a regalarti un sorriso e uno svago nel momento del bisogno.*

## 7. RINGRAZIAMENTI

---

*A Lorenzo, un capo prima, un amico adesso, che nel momento in cui tutto poteva andare perduto, è riuscito con un cartellino a salvare una persona in difficoltà, e a renderla davvero migliore.*

*A WAYouth, non una community, non una associazione, non solo una famiglia, ma il luogo in cui poter essere se stessi liberamente, senza pregiudizi e senza etichette. Il luogo in cui un giovane ragazzo della provincia di Napoli è cresciuto, ha messo la testa apposto ed ora si può dire veramente orgoglioso di se stesso.*

*A Raffaele e a Pietro, compagni di questa avventura, anche loro sempre pronti a coprirmi e ad aiutarmi nel momento del bisogno.*

*A Salvatore, amico fraterno, colonna sonora di questo viaggio.*

*A Rocco, a Roberto, a Delfina, a Giovanni, ad Alberto, a Rosalba, a Clelia, a Patrizia, a Vincenzo, ad Andrea, ad Adele, a Salvatore, a Michele, a Genoveffa, a Ugo, ad Antonio, a Francesco, a Filomena, a Carmine, a Luisa, al compianto Ignazio, ad Alfredo. A tutti voi, a chi ho voluto bene, e a chi ho odiato, un'enorme grazie per il viaggio e il sapere tramandato.*

*A tutta la 5Di e a tutta la loro cazzimma, per esserci sempre stati.*

*A Diamante, a tutti gli scappiati di testa di quel gruppo, sempre pronti a dirmi "non ti fai ne vedere e ne sentire", ma sempre lì ogni volta che mi giro.*

*A tutte le persone che ho incontrato lungo il mio percorso, belle o brutte, simpatiche o antipatiche, che ho lasciato dopo due secondi, o che mi seguono da allora. Grazie ad ognuno di voi sono qui dove sono.*

*A me stesso.*

---

## BIBLIOGRAFIA

- [1] <https://www.r3m.it/MICHAEL-JACKSON-LA-STORIA-ASSURDA-DEL-PRESUNTO-PLAGIO-AD-ALBANO/>
- [2] <https://www.r3m.it/BEATLES-QUELLA-VOLTA-CHE-GEORGE-HARRISON-FU-ACCUSATO-DI-PLAGIO-PER/>
- [3] <http://www.youtube.com/T/CONTENTID>
- [4] COMMENTO TRATTO DA “BREVI NOTE SUL PLAGIO MUSICALE” (TRATTO DALLE LEZIONI PRESSO L’UNIVERSITÀ DI GIURISPRUDENZA DI PADOVA) ARCHIVIATO IL 29 DICEMBRE 2009 IN INTERNET ARCHIVE. <HTTPS://WEB.ARCHIVE.ORG/WEB/20091229050239/HTTP://WWW.DELLARTEGAMBINO.IT/CONTENTS/ALLEGATI/20050502-NOTE>
- [5] ANDREA SIROTTI GAUDENZI, “IL NUOVO DIRITTO D'AUTORE”, 2009
- [6] PRINCE ROGER NELSON, CONTROVERSY MUSIC INC., MICHELE VICINO, BRUNO BERGONZI, [HTTPS://WWW.DIRITTODAUTORE.IT/WP-CONTENT/UPLOADS/2015/06/SENTCASSCIV11225\\_2015.PDF](HTTPS://WWW.DIRITTODAUTORE.IT/WP-CONTENT/UPLOADS/2015/06/SENTCASSCIV11225_2015.PDF)

---

## BIBLIOGRAFIA

---

- [7] R. DE PRISCO, A. ESPOSITO, N. LETTIERI, D. MALANDRINO, D. PIROZZI, G. ZACCAGNINO, AND R. ZACCAGNINO, “MUSIC PLAGIARISM AT A GLANCE: METRICS OF SIMILARITY AND VISUALIZATIONS,” IN2017 21ST INTERNATIONAL CONFERENCE INFORMATIONVISUALISATION (IV). IEEE, 2017, PP. 410–415.
- [8] [HTTPS://WWW.IDMT.FRAUNHOFER.DE/CONTENT/DAM/IDMT/DOCUMENTS/IL/MUSIC\\_PLAGIARISM\\_DETECTION\\_EN.PDF](HTTPS://WWW.IDMT.FRAUNHOFER.DE/CONTENT/DAM/IDMT/DOCUMENTS/IL/MUSIC_PLAGIARISM_DETECTION_EN.PDF)
- [9] W. H. GOMAA, A. A. FAHMYET AL., “A SURVEY OF TEXT SIMILARITY APPROACHES,” INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS, VOL. 68, NO.13,2013.  
[HTTPS://WWW.ACADEMIA.EDU/9531934/A\\_SURVEY\\_OF\\_TEXT\\_SIMILARITY\\_APPROACHES](HTTPS://WWW.ACADEMIA.EDU/9531934/A_SURVEY_OF_TEXT_SIMILARITY_APPROACHES)
- [10] ERIC SVEN RISTAD, PETER N. YIANILOS, LEARNING STRING EDIT DISTANCE, 1997. <HTTP://CITESEERX.IST.PSU.EDU/VIEWDOC/SUMMARY?DOI=10.1.1.39.3604>
- [11] MILLER, G.A., BECKWITH, R., FELLBAUM, C.D., GROSS, D. & MILLER, K. (1990). WORDNET: AN ONLINE LEXICAL DATABASE. INT. J. LEXICOGRAPH. 3, 4, PP. 235-244.
- [12] R. FERRAIOLI, ”AN ADAPTIVE META-HEURISTIC FOR MUSIC PLAGIARISM DETECTION: TEXT SIMILARITY AND MULTI-OBJECTIVE OPTIMIZATION”, TESI DI LAUREA MAGISTRALE , 2019
- [13] NGATCHOU, A. ZAREI, AND A. EL-SHARKAWI, “PARETO MULTI OBJECTIVE OPTIMI-ZATION,” INPROCEEDINGS OF THE 13TH INTERNATIONAL CONFERENCE ON, INTELLIGENTSYSTEMS APPLICATION TO POWER SYSTEMS. IEEE, 2005, PP. 84–91

## BIBLIOGRAFIA

---

- [14] "FLASK FOREWORD". ARCHIVED FROM THE ORIGINAL ON 2017-11-17. [HTTPS://WEB.ARCHIVE.ORG/WEB/20171117015927/HTTP://FLASK.POCOO.ORG/DOCS /0.10/FOREWORD](https://web.archive.org/web/20171117015927/http://flask.pocoo.org/docs/0.10/foreword)
- [15] "FLASK EXTENSIONS". ARCHIVED FROM THE ORIGINAL ON 2018-05-17. [HTTPS://WEB.ARCHIVE.ORG/WEB/20180517082208/HTTP://FLASK.POCOO.ORG/EXTENSIONS/](https://web.archive.org/web/20180517082208/http://flask.pocoo.org/extensions/)
- [16] WHAT CHALLENGES HAS PINTEREST ENCOUNTERED WITH FLASK? [HTTPS://WWW.QUORA.COM/WHAT-CHALLENGES-HAS-PINTEREST-ENCOUNTERED-WITH-FLASK/ANSWER/STEVE-COHEN?RID=HXZD&SHARE=1](https://www.quora.com/What-challenges-has-Pinterest-encountered-with-Flask/answer/Steve-Cohen?rid=HXZD&share=1)
- [17] RACHEL SANDERS: DEVELOPING FLASK EXTENSIONS - PYCON 2014  
[HTTPS://WWW.YOUTUBE.COM/WATCH?V=OXN3WUHUBP0#T=46](https://www.youtube.com/watch?v=oxN3WUHUBP0#t=46)
- [18] A. DEL GAIZO, "AN ADAPTIVE META-HEURISTIC FOR MUSIC PLAGIARISM DETECTION: WORD EMBEDDING AND CLUSTERING", TESI DI LAUREA MAGISTRALE, 2019
- [19] [HTTPS://WWW.JETBRAINS.COM](https://www.jetbrains.com)
- [20] L. CIARAVOLA, "SVILUPPO DI UNA PIATTAFORMA WEB PER IL RILEVAMENTO E LA GESTIONE DEL PLAGIO MUSICALE", TESI DI LAUREA, 2021
- [21] R. SQUILLANTE, "SVILUPPO DI UNA APPLICAZIONE ANDROID PER IL RILEVAMENTO E LA GESTIONE DEL PLAGIO MUSICALE", TESI DI LAUREA, 2021