

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Tesi di laurea di I livello in
Informatica

**Sviluppo di una applicazione Android
per il rilevamento e la gestione del
plagio musicale**

Primo Relatore:

Rocco Zaccagnino

Secondo Relatore:

Roberto De Prisco

Candidato:

Raffaele Squillante
Mat. 05121 06093

ANNO ACCADEMICO 2020/2021

*“I computer sono incredibilmente veloci, accurati e stupidi.
Gli uomini sono incredibilmente lenti, inaccurati e intelligenti.
L’insieme dei due costituisce una forza incalcolabile.”*

INDICE

1 ABSTRACT	5
2 INTRODUZIONE	7
2.1 Il 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
2.3 Obiettivo della tesi	10
2.4 Organizzazione della tesi	11
3 FASE DI RICERCA	12
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

4 Fase di studio tecnologico	16
4.1 Funzionalità legacy riutilizzate	16
4.1.1 Metriche utilizzate nel sistema legacy	16
4.1.2 MasS – 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-Bcrypt	21
4.3.3 Flask-Mail	21
4.4 Java per Android	21
4.5 XML	22
4.6 JSON	22
5 Fase di progettazione	24
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	27
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
6 FASE DI SVILUPPO	31
6.1 Tecnologie di implementazione	31
6.1.1 Activity e Fragment	31
6.1.2 Icone utilizzate nei casi di plagio	32
6.1.3 Permessi Android	33

INDICE

6.1.4	Tipi di permission	35
6.1.5	Gestione delle permission	36
6.2	Ambienti di sviluppo utilizzati	37
6.2.1	PyCharm	37
6.2.2	Android Studio	37
6.2.3	Adobe XD	38
6.2.4	Adobe Photoshop	39
6.3	Fasi di implementazione	39
6.3.1	Definizione della struttura del progetto	39
6.3.2	Inizializzazione e configurazione del progetto	39
6.3.3	Sviluppo del layout	41
6.4	SubSystem Main	42
6.4.1	Caricamento brani	42
6.4.2	Visualizzazione e schematizzazione dei risultati	43
6.5	SubSystem Users	44
6.5.1	Autenticazione	44
6.5.2	Aggiornamento informazioni dell'account	45
6.6	SubSystem Sentences	47
6.6.1	Visualizzazione casi di plagio	47
6.6.2	Visualizzazione singolo caso di plagio	47
6.6.3	Ricerca casi di plagio	49
6.6.4	Inserimento caso di plagio	50
6.7	Sviluppi futuri	50
7	RINGRAZIAMENTI	52

CAPITOLO 1

ABSTRACT

Il plagio musicale, l'atto di copiare un brano o parte di esso da altri compositori, è diventato un aspetto estremamente importante 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.

Al giorno d'oggi però non vi è ancora nessuna applicazione mobile in commercio capace di offrire tali strumenti al pubblico.

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 estendendo le funzionalità di una piattaforma web. 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 metaeuristica adattiva basata su text similarity e Clustering; fornendo inoltre funzionalità di inserimento e gestione di plagio all'interno della base

1. ABSTRACT

dati.

Il mio lavoro di tesi, principalmente, si basa sul problema di creare un layout grafico e successiva implementazione delle funzionalità offerte dal server legacy sulla nostra applicazione, più in generale, su tutta la realizzazione del front-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 Il 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.

2. INTRODUZIONE

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 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].

2. INTRODUZIONE

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].

2.3 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, 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 legacy¹ 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 analytics², i singoli casi di plagio e poter così procedere con decisioni ponderate. Tale piattaforma legacy, infatti, permette di calcolare i dati del confronto tra due brani in input mediante l’utilizzo di algoritmi di text similarity e data clustering applicati ai brani. Sfruttando le funzionalità della piattaforma, l’oggetto del lavoro di tesi si basa sul rendere fruibili tali servizi, mediante App dedicata, su dispositivi Mobile Android, in particolare il mio lavoro di tesi si basa sul problema di creare un layout grafico e successiva implementazione delle funzionalità offerte dal server legacy sulla nostra applicazione, più in generale, su tutta la realizzazione del front-end.

¹Per “Sistema Legacy” o “Progetto Legacy”, ci riferiamo alla precedente applicazione, basata sui lavori di R. Ferraioli, A. Del Gaizo e L. Ciaravola. [12]/[18]/[20]

²Tradotto 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.

2.4 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 dell'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.
- **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 di ricerca 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 da utilizzare dal sistema legacy. La maggior parte dell'attività è stata dedicata agli algoritmi di Text-Similarity.
- Nella terza sottofase ci siamo focalizzati sulla ricerca di Applicazioni 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 corrispondano 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 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 nuova 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 motivi simili o identici.
- Utilizzare algoritmi specializzati per valutare automaticamente il grado di similarità tra le 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 fa 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 di 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 frasi, 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 sfrutta 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 Corpus-Based è una misura di similarità semantica che determina la similarità tra parole in base ad informazioni ottenute da grosse corpora¹. 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 sfruttando informazioni derivate da network semantici. Un esempio di network semantico è WordNet [11].

¹*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 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

Dal lavoro precedente abbiamo estrapolato diverse funzionalità, adattando e aggiornando 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)
- Dice's coefficient (DC)
- Jaro similarity (JRS)

4. FASE DI STUDIO TECNOLOGICO

- Overlap Coefficient (OC)

Di seguito illustriamo brevemente ciascuna metrica utilizzata nel sistema legacy, omettendo le rappresentazioni matematiche poiché 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 0 che i due documenti non si somigliano affatto.

- **Cosine similarity**

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

- **Dice's coefficient**

Noto anche come indice Sorensen-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 stringhe sono uguali e 0 significa che non è riscontrata alcuna somiglianza tra le due stringhe.

4.1.2 MasS – Music as String

Le funzionalità MasS (“*Music as String*”) del progetto legacy, utilizzate per la conversione di brani xml in una stringa testuale utilizzata per la rilevazione del plagio, sono state inglobate in due metodi che si occupano di differenti rappresentazioni:

- la prima rappresentazione è utilizzata per la rilevazione del plagio.
- la seconda rappresentazione è utilizzata per il calcolo della sottosequenza più' lunga di note in comune.

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 queste metriche di text similarity ogni volta.

4. FASE DI STUDIO TECNOLOGICO

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 similarity text based 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 tra i risultati calcolati nella nostra applicazione Android)

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-Bcrypt

Flask-Bcrypt è un'estensione Flask che fornisce delle *utilities* di hashing bcrypt per le applicazioni, utilizzata per la protezione di dati sensibili. Nel nostro progetto è utilizzato per l'hashing delle password, permettendo la trasformazione della password inserita dall'utente in una stringa senza significato apparente; questa sarà la stringa che salveremo nel database, evitando di salvare le password dell'utente in chiaro. All'autenticazione dell'utente si utilizzano le funzionalità di verifica di Flask Bcrypt per confrontare la password inserita con la stringa salvata nel database.

4.3.3 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 poiché 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 programmatore 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 programmatore 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:

4. FASE DI STUDIO TECNOLOGICO

- 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.

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 descrivere 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.

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

5. FASE DI PROGETTAZIONE

nell'applicazione. 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. FASE DI PROGETTAZIONE

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 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* appoggiandosi alla

5. FASE DI PROGETTAZIONE

sottostante architettura protocollare.

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 prima in forma temporanea per il confronto corrente, per poi essere gestito dal sottosistema Sentences in caso di inserimento nel sistema.

Funzionalità:

- Caricamento brani
- Calcolo dei valori del confronto
- Salvataggio dei dati del confronto corrente
- Visualizzazione e schematizzazione dei risultati

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, 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; l’aggiornamento dei *dataset* viene effettuato ogni volta che viene inserito un nuovo caso, aggiornati i brani di un caso precedente o eliminato un vecchio caso.

Funzionalità:

5. FASE DI PROGETTAZIONE

- Visualizzazione casi di plagio
- Visualizzazione dati del singolo caso di plagio
- Ricerca casi di 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 registrazione di un nuovo utente tecnico, 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
- Reset password dell'account
- Aggiornamento informazioni dell'account

5.4 Tipologie di utenti

Al momento la piattaforma considera due tipologie di utenti:

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

5. FASE DI PROGETTAZIONE

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

In questo capitolo verranno descritte le 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

6. FASE DI SVILUPPO

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 Icone utilizzate nei casi di plagio

L'applicazione utilizza delle coppie di icone associate ai casi di plagio per comunicare visivamente all'utente se il caso è considerato un plagio o meno, per comunicare lo stato giuridico del caso di plagio e per segnalare la presenza di un verdetto in formato PDF:

- **Icone Plagio:**



Secondo il sistema, il caso con questa icona è considerato Plagio.



Secondo il sistema, il caso con questa icona non è considerato Plagio.

- **Icone Stato Giuridico:**



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 ed esiste quindi un verdetto.

- Icône presenza del verdetto :



Il caso con questa icona permette il download del verdetto associato.

6.1.3 Permessi Android

Un'app Android vive in all'interno di una *sandbox*, ovvero un ambiente chiuso in cui l'app opera in maniera sostanzialmente isolata dal resto del sistema. Talvolta, però, può essere necessario che essa debba “uscire” da questa gabbia per accedere a informazioni, funzionalità o apparati hardware del dispositivo. Per far sì che l'app acceda a tali servizi, essa deve possedere alcuni “permessi” speciali (*permission*) che è necessario dichiarare espressamente nel file di configurazione `AndroidManifest.xml` (Figura 6.1). Ogni permission viene dichiarata tramite il tag `uses-permission`, da collocare al di fuori del nodo `application`.

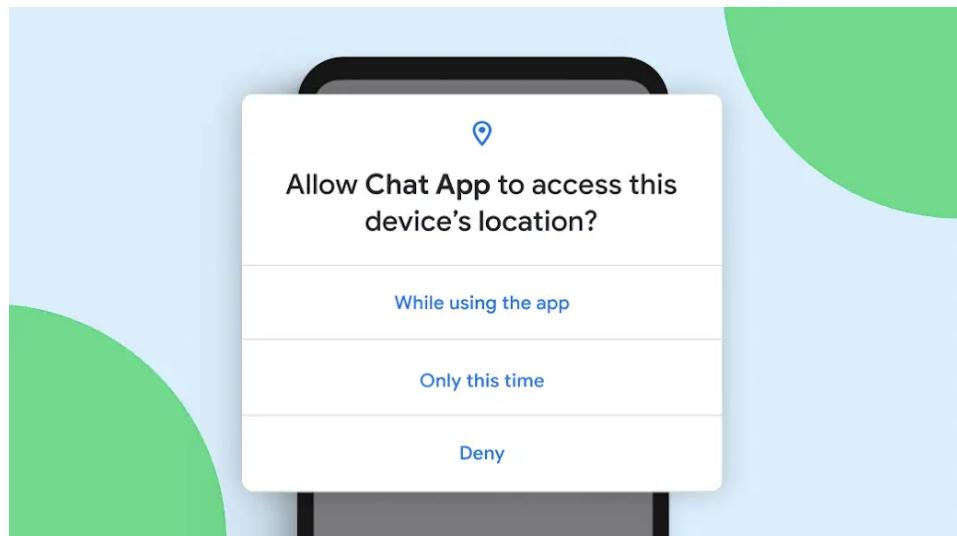
Figura 6.1: Esempio di `AndroidManifest.xml`

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

Fatto ciò, il sistema operativo saprà quali attività particolari l'app dovrà svolgere, e potrà concederle il permesso di effettuarle. Prima dell'installazione dell'app, infatti, l'utente viene sempre avvisato delle permission richieste dall'app: se accetta di installarla, accetta che il suo sistema operativo conceda tali permessi “speciali” all'app (Figura 6.2).

6. FASE DI SVILUPPO

Figura 6.2: Esempio di *Permission*



6.1.4 Tipi di permission

Le permission Android vengono suddivise in tre famiglie in base al livello di protezione:

- **Permission normal:** che non mettono cioè a rischio la privacy dell’utente, tra le quali troviamo:

- **Accesso alla Rete:** permette all’applicazione l’accesso alla rete ed effettuare richieste http e ricevere risposte
`android.permission.INTERNET`
`android.permission.ACCESS-NETWORK-STATE`
`android.permission.ACCESS-WIFI-STATE`

- **Accesso a particolari tecnologie per la connettività:** consente all’applicazione di poter utilizzare ad esempio *Bluetooth* o *NFC*
`android.permission.BLUETOOTH`
`android.permission.NFC`

- **Invio di un Intent per l’impostazione di un allarme:** consente all’applicazione di settare custom intent
`com.android.alarm.permission.SET-ALARM`
 - **Gestione della vibrazione:** consente all’applicazione di gestire la vibrazione
`android.permission.VIBRATE`

- **Permission dangerous:** potenzialmente lesive per la riservatezza degli utenti, come quelle riguardanti:

- **La localizzazione:** consente all’applicazione di accedere al sensore *GPS*
`android.permission.FINE-LOCATION`
`android.permission.ACCESS-COARSE-LOCATION`

- **Il calendario:** consente all’applicazione di leggere e scrivere sul calendario

6. FASE DI SVILUPPO

android.permission.READ-CALENDAR

android.permission.WRITE-CALENDAR

– **I contatti:**

consente all'applicazione di leggere e modificare la lista dei contatti

android.permission.READ-CONTACTS

android.permission.WRITE-CONTACTS

– **Attività telefoniche:**

consente all'applicazione di leggere le attività telefoniche come sms
e ottenere stato della linea

android.permission.READ-PHONE-STATE

android.permission.SEND-SMS

– **Accesso ai file:** consente all'applicazione di poter leggere e scrivere
sullo storage dedicato

android.permission.READ-EXTERNAL-STORAGE

android.permission.WRITE-EXTERNAL-STORAGE

- **Permission signature:** gestite a livello di installazione ma utilizzabili
solo se l'app che le richiede è firmata con lo stesso certificato di quella
che ha definito la permission.

6.1.5 Gestione delle permission

Le permission *dangerous* vanno accettate a runtime dall'utente al
momento dell'utilizzo delle funzionalità che le richiedono, mentre per quelle
normal è sufficiente dichiararne l'impiego al momento dell'installazione
dell'app. Tale distinzione è stata introdotta in Android 6 (API 23) ed è ormai
prassi cui l'utente è totalmente abituato. Inoltre, una permission dangerous
già concessa potrà essere revocata in qualsiasi istante.

6.2 Ambienti di sviluppo utilizzati

Per lo sviluppo dell'applicazione abbiamo utilizzato *PyCharm* e *Android Studio*, mentre per realizzare l'intero aspetto grafico abbiamo utilizzato *Adobe XD* e *Adobe Photoshop*.

6.2.1 PyCharm

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
- Graphic Debugger
- Unit Tester integrat.
- 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

Android Studio è anch'esso un ambiente di sviluppo integrato (IDE) che utilizza diversi linguaggi di programmazione quali Kotlin, Java, and C/C++, ed è progettato specificamente per lo sviluppo di applicazioni Android. Per la nostra app abbiamo deciso di utilizzare Java come linguaggio di programmazione, in quanto è uno dei linguaggi più utilizzati e completi al momento. Androdi Studio mette a disposizione vari strumenti come:

- Visual layout editor
- APK Analyzer
- Fast emulator
- Intelligent code editor
- Realtime profilers

Importantissimo è l'emulatore gestito tramite la componente *AVD* (Android Virtual Device) che permette di testare l'applicazione su dispositivi con diverse caratteristiche, emulati, evitando di dover possedere più dispositivi o rilasciare un'applicazione prematuramente.

6.2.3 Adobe XD

Adobe XD è un software di progettazione dell'esperienza utente per app web e app mobili basato su grafica vettoriale, prodotto e distribuito da Adobe all'interno della raccolta Creative Cloud. È disponibile per macOS e Windows, anche se ci sono versioni per iOS e Android per aiutare a visualizzare in anteprima il risultato del lavoro direttamente sui dispositivi mobili. Adobe XD permette di:

- Aggiungere profondità e prospettiva agli oggetti
- Creare pulsanti riutilizzabili e schede ridimensionabili
- Aggiungere stati, interazioni ed eventi ai componenti creati
- Duplicare intere gallerie ed elenchi.
- Apportare modifiche all'istante con semplici click sugli elementi

Adobe XD permette di creare interi layout in modo facile e veloce, dando libero spazio alla propria creatività grazie all'ampio bacino di *tool* utilizzabili. Inoltre con Adobe XD è possibile esportare i singoli componenti creati sia in formato vettoriale SVG, sia, dove non fosse possibile, in un formato PNG progettato appositamente per Android per ottenere la massima qualità possibile.

6.2.4 Adobe Photoshop

Adobe Photoshop è un software proprietario prodotto da Adobe specializzato nell’elaborazione di fotografie e, più in generale, di immagini digitali. Questo programma è in grado di effettuare ritocchi di qualità professionale alle immagini, offrendo enormi possibilità creative grazie ai numerosi filtri e strumenti che permettono di emulare le tecniche utilizzate nei laboratori fotografici per il trattamento delle immagini, le tecniche di pittura e di disegno. Un’importante funzione del programma è data dalla possibilità di lavorare con più livelli, permettendo di gestire separatamente le diverse componenti che costituiscono l’immagine principale.

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 *front-end* e *back-end* in accordo con la scomposizione modulare definita in fase di progettazione. Per questo lavoro di tesi ci concentreremo sul lato front-end, mentre il back-end sarà trattato dal mio collega Giametta Antonio [21].

6.3.2 Inizializzazione e configurazione del progetto

Per prima cosa è stato necessario installare e configurare PyCharm, scaricando tutti i moduli necessari, per poter utilizzare il server legacy. Successivamente è stato creato un nuovo progetto su Android Studio dove sono state create le **Activity** relative alle view, contenenti le funzionalità da sviluppare. Abbiamo deciso di utilizzare una **BottomNavigationView** come menù di pagina e quindi abbiamo dovuto implementare tutte le view contenenti quest’ultimo come **Fragment**. Infine è stato descritto il file di configurazione **AndroidManifest.xml** (Figura 6.3) contenente le informazioni principali dell’applicazione. In particolare l’app può essere usata solo in

6. FASE DI SVILUPPO

modalità *portrait* e per essere utilizzabile ha bisogno dei permessi di accesso ad Internet e di lettura e scrittura sullo storage.

Figura 6.3: *AndroidManifest.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.plagiomusicale">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        tools:ignore="ScopedStorage" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

    <application
        android:windowSoftInputMode="stateVisible|adjustPan"
        android:hardwareAccelerated="false"
        android:usesCleartextTraffic="true"
        android:networkSecurityConfig="@xml/network_security_config"
        android:allowBackup="true"
        android:icon="@drawable/icona"
        android:label="@string/app_name"
        android:roundIcon="@drawable/icona"
        android:supportsRtl="true"
        android:theme="@style/AppTheme.NoActionBar"
        tools:ignore="AllowBackup">
        <activity android:name=".Login"
            android:windowSoftInputMode="stateVisible|adjustPan"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".About"
            android:windowSoftInputMode="stateVisible|adjustPan"
            android:screenOrientation="portrait">
        </activity>
        <activity android:name=".Menu"
            android:windowSoftInputMode="stateVisible|adjustPan"
            android:screenOrientation="portrait">
        </activity>
    </application>

</manifest>
```

6.3.3 Sviluppo del layout

L'intero layout è stato realizzato con l'uso di Adobe XD e Adobe Photoshop e successivamente è stato incorporato al layout XML di Android Studio. Sono stati realizzati tre layout di base:

- Uno per le activity (Figura 6.4)
- Uno per i fragment con menù per *guest* (Figura 6.6)
- Uno per i fragment con menù per gli utenti loggati (Figura 6.5)

Il layout è stato realizzato seguendo dei corsi/tutorial di *modern design* per ricercare un design *minimal, moderno* ed *accattivante* che potesse risaltare il valore dell'applicazione.

Figura 6.4: *Background Activity*

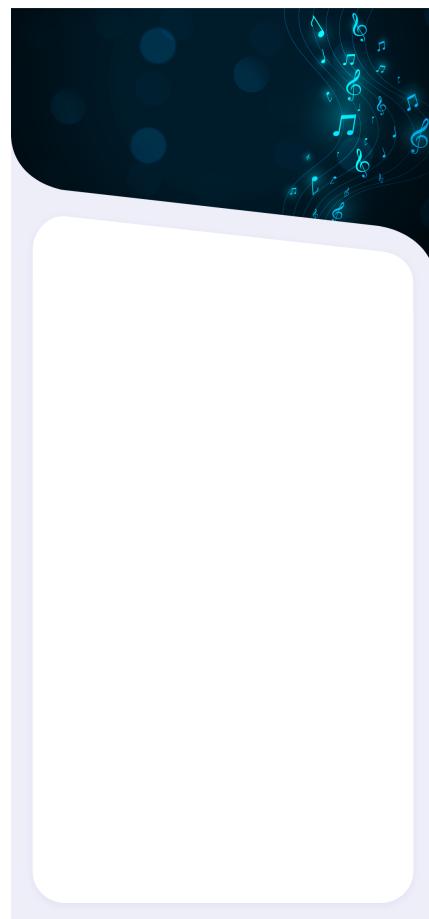


Figura 6.5: *Fragment Guest*



Figura 6.6: *Fragment Utente Loggato*



Di seguito sono descritte tutte le funzionalità che sono state realizzate, seguendo i sottosistemi descritti nella fase di progettazione.

6.4 SubSystem Main

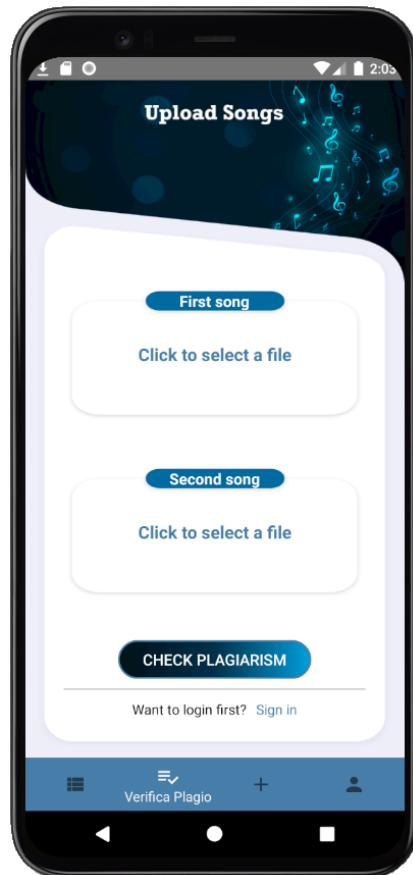
6.4.1 Caricamento brani

La funzionalità di caricamento dei brani è stata implementata tramite il Fragment `UploadSong` (Figura 6.7), contenente i `RelativeLayout` rappresentanti i brani, ai quali sono stati imposti dei vincoli sui formati caricabili, restringendo il caricamento dei files al solo formato XML.

6. FASE DI SVILUPPO

Una volta caricati i brani, il Fragment crea una connessione con il server e chiama la route `upload_songs_Mobile()`, appositamente modificata per lo scambio dei dati con l'applicazione Mobile, che si occupa di verificare se sussiste o meno il plagio. Al termine della verifica, i brani vengono salvati nelle cartelle legacy e nel sistema come parte dei dati del confronto corrente, reindirizzando l'utente alla pagina di visualizzazione dei risultati.

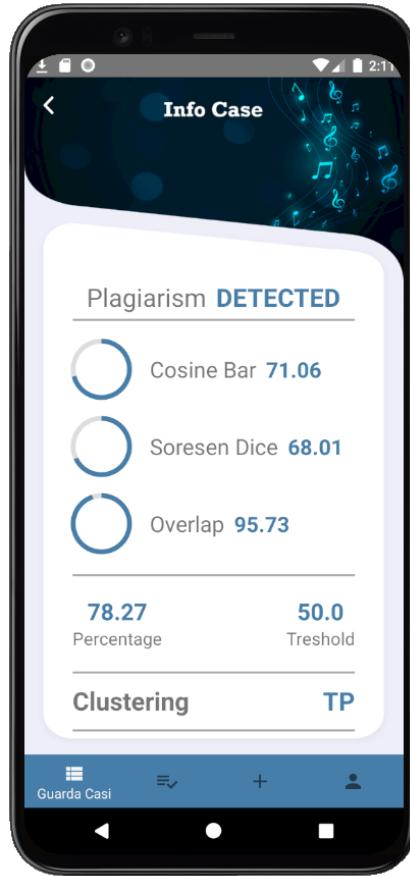
Figura 6.7: *Upload Song*



6.4.2 Visualizzazione e schematizzazione dei risultati

La visualizzazione dei risultati è gestita dal Fragment `ResultsCheck` (Figura 6.8), che si occupa della formattazione e della visualizzazione dei valori precedentemente calcolati e dei nomi delle metriche utilizzate.

Figura 6.8: *ResultsCheck*



6.5 SubSystem Users

6.5.1 Autenticazione

La funzionalità di autenticazione è implementata con l'Activity `Login` (Figura 6.9), anche `MainActivity` dell'applicazione, che permette l'accesso utilizzando email e password. Essa chiama la route `loginMobile()` si occupa dei controlli di login seguenti:

- Controlla che l'utente sia presente nel database tramite l'email inserita
- Effettua l'hashing ed il controllo della password sfruttando le funzionalità di Flask-Bcrypt [vedi Flask-Bcrypt 3.4.3]

6. FASE DI SVILUPPO

Se i controlli hanno esito positivo l'utente verrà reindirizzato alla pagina di visualizzazione dei casi di plagio.

Se l'utente non è registrato può comunque continuare ad usare l'applicazione effettuando il login come guest. In questo caso l'utente avrà a disposizione funzionalità limitate. [vedi Tipologie di utenti 5.4]

Da questa pagina è anche possibile accedere all'Activity **AboutUs** (Figura 6.10), che contiene una piccola descrizione dell'applicazione.

Figura 6.9: *Login*

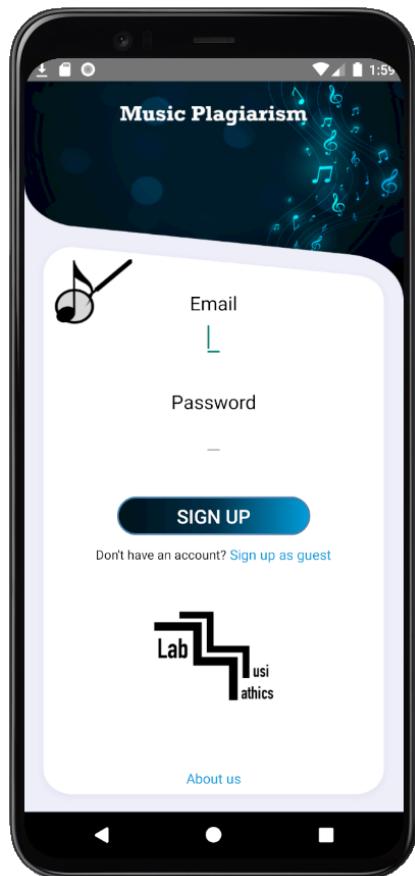
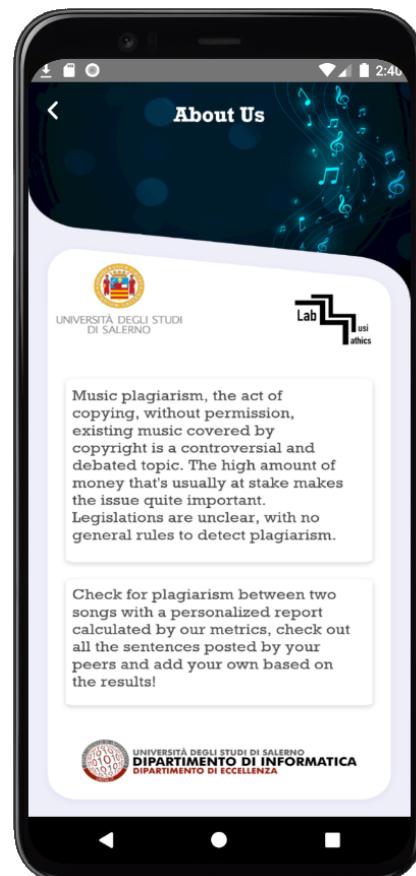


Figura 6.10: *AboutUs*



6.5.2 Aggiornamento informazioni dell'account

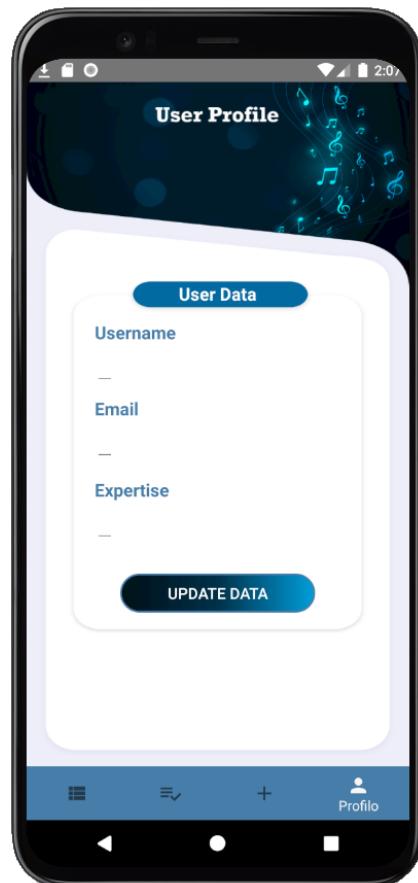
La funzionalità di aggiornamento account è stata implementata nel Fragment **UserPage** (Figura 6.11), che:

6. FASE DI SVILUPPO

- Chiama la route `accountMobile()` per ottenere tutti i dati che l'utente visualizzerà sulla pagina profilo
- Chiama la route `accountUpdateMobile()` per l'aggiornamento dei dati dell'account

Qui l'utente può cambiare o lasciare inalterate alcune informazioni personali che, alla pressione del Button "Update Data" verranno aggiornate nel database. Dopodiché l'utente viene reindirizzato alla pagina di account, dove visionerà i dati aggiornati.

Figura 6.11: *UserPage*

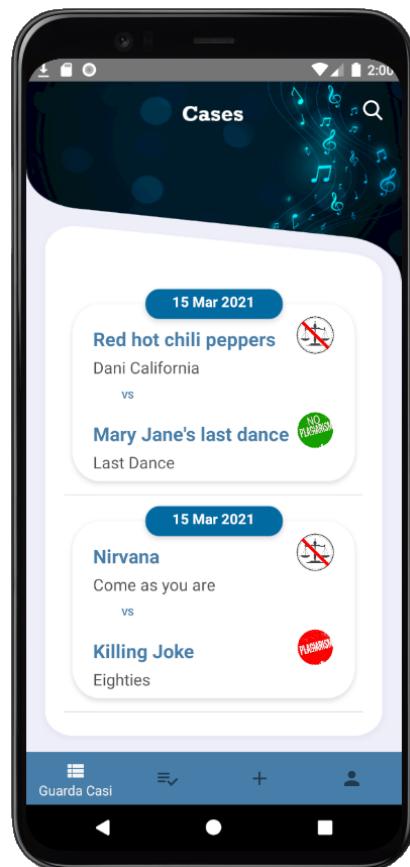


6.6 SubSystem Sentences

6.6.1 Visualizzazione casi di plagio

La funzionalità di visualizzazione dei casi di plagio è implementata nel Fragment `ViewCases` (Figura 6.12) che, tramite un `ListView`, un `CustomAdapter` ed un Layout appositamente creato per ogni sentenza, mostra all'utente tutti i casi presenti sul database chiamando la route `all_sentencesMobile()`.

Figura 6.12: *ViewCases*



6.6.2 Visualizzazione singolo caso di plagio

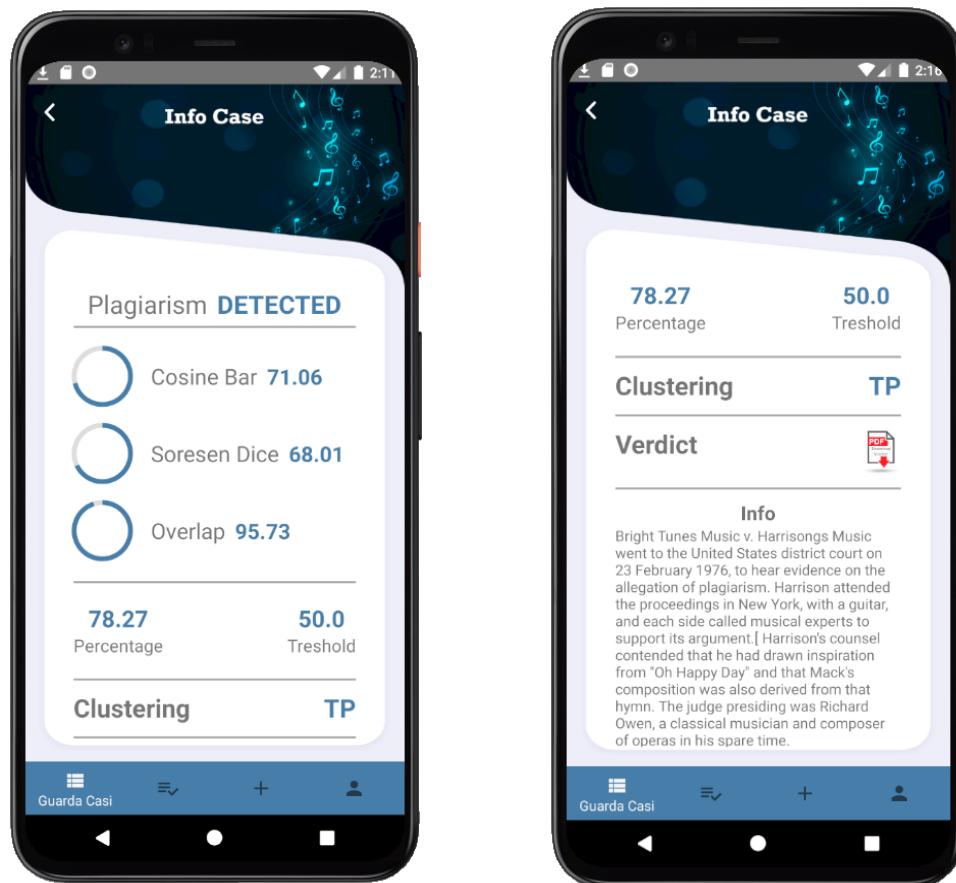
La visualizzazione di un singolo caso di plagio è implementata tramite il Fragment `Results` (Figura 6.13) che, sfruttando la posizione del caso nel `ListView`, recupera i dati relativi dal database e dai file associati, per mostrarli

6. FASE DI SVILUPPO

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

- Esito del calcolo del plagio
- Metriche e valori utilizzati nel calcolo del plagio
- Informazioni aggiuntive
- visualizzazione del PDF del verdetto scaricabile (in caso di verdetto non presente, neanche il campo "verdetto" sarà presente)

Figura 6.13: *Results*



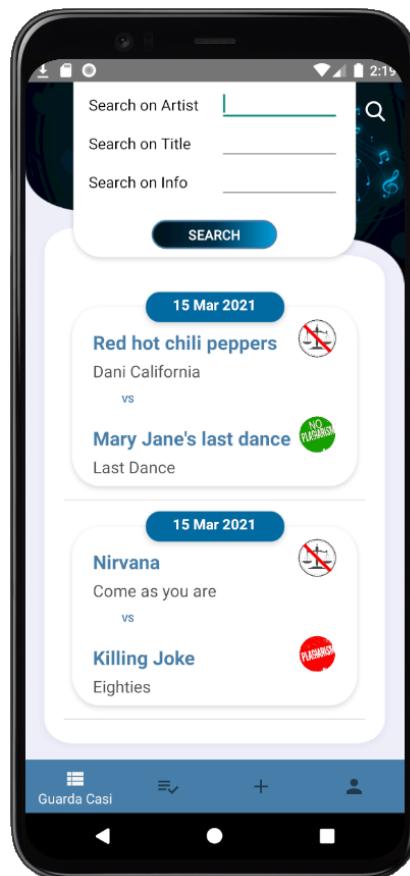
6.6.3 Ricerca casi di plagio

La funzionalità di ricerca è implementata tramite un menù a tendina presente nel Fragment **ViewCase** (Figura 6.14), e accessibile tramite apposita icona, che permette l'inserimento di valori in tre campi:

- Titolo del primo brano
- Titolo del secondo brano
- Informazioni contenute nel campo info

Alla pressione del Button **Search**, sfruttando i metodi di SQLAlchemy, carica dal database una lista di casi che rispettano i filtri inseriti e aggiorna la pagina mostrando solo i casi che hanno riscontrato un *matching*. I campi permettono l'inserimento completo o parziale dei titoli e delle informazioni aggiuntive.

Figura 6.14: *Search*

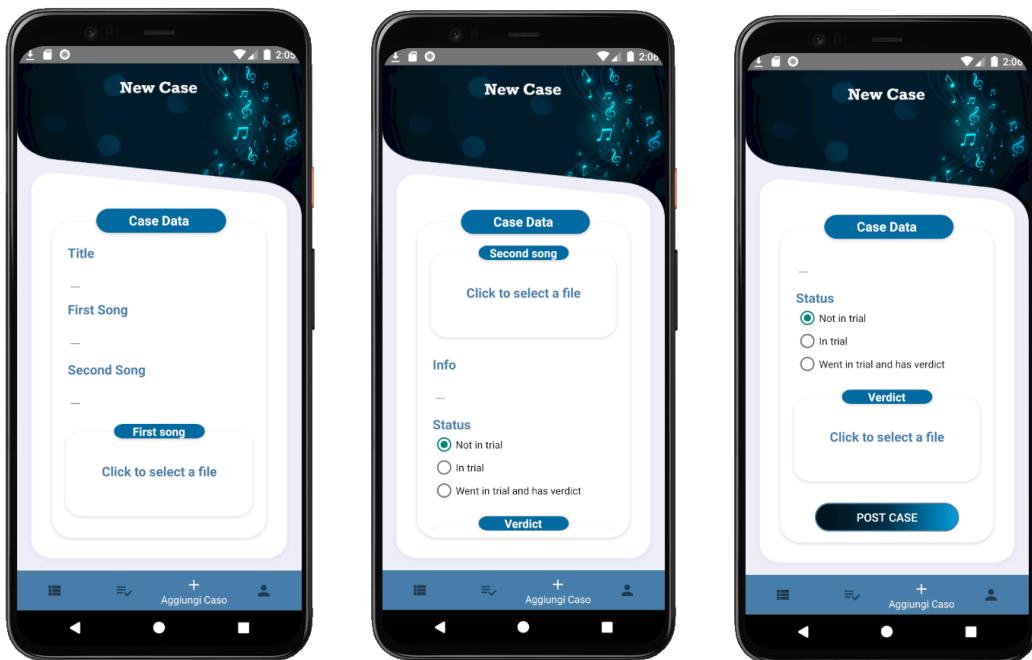


6.6.4 Inserimento caso di plagio

L'inserimento di casi di plagio, funzionalità per utenti tecnici registrati, si può effettuare tramite il Fragment **NewCase** (Figura 6.15).

L'inserimento dei dati avviene tramite appositi campi che guidano l'utente nella compilazione della sentenza. Ad inserimento effettuato la route del sistema procede all'inserimento del caso di plagio nel database.

Figura 6.15: *NewCase*



6.7 Sviluppi futuri

In futuro si potrebbe migliorare la piattaforma 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.

6. FASE DI SVILUPPO

- Un aggiornamento o ampliamento della metaeuristica sulla quale si basa la rilevazione del plagio, inserendo magari la possibilità di aggiungere e 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

Un ringraziamento speciale va ai miei genitori, senza di voi niente di tutto ciò sarebbe stato possibile.

Ringrazio tutti i miei amici, quelli che ci sono da sempre e quelli che ho conosciuto e che mi hanno seguito in questa fantastica avventura.

Ringrazio la mia famiglia, che mi ha supportato in questo percorso.

Ultimo, ma non per importanza, un ringraziamento speciale ad una persona speciale che avrebbe voluto tanto esserci oggi... Grazie nonna.



BIBLIOGRAFIA

- [1] <HTTPS://WWW.R3M.IT/MICHAEL-JACKSON-LA-STORIA-ASSURDA-DEL-PRESUNTOPLAGIO-AD-ALBANO/>
- [2] <HTTPS://WWW.R3M.IT/BEATLES-QUELLA-VOLTA-CHE-GEORGE-HARRISON-FUACCUSATO-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%20SUL%20PLAGIO%20MUSICALE.PDF>
- [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

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
- [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
- [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] P. NGATCHOU, A. ZAREI, AND A. EL-SHARKAWI, “PARETO MULTI OBJECTIVE OPTIMIZATION,” 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>
- [15] "FLASK EXTENSIONS". ARCHIVED FROM THE ORIGINAL ON 2018-05-17.
<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?SRID=HXZD&SHARE=1>
- [17] RACHEL SANDERS: DEVELOPING FLASK EXTENSIONS - PYCON 2014 <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/>
- [20] L. CIARAVOLA, "SVILUPPO DI UNA PIATTAFORMA WEB PER IL RILEVAMENTO E LA GESTIONE DEL PLAGIO MUSICALE", TESI DI LAUREA TRIENNALE, 2021
- [21] A. GIAMETTA, "SVILUPPO DI UNA APPLICAZIONE ANDROID PER IL RILEVAMENTO E LA GESTIONE DEL PLAGIO MUSICALE", TESI DI LAUREA TRIENNALE, 2021