Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

(DIEM)

Corso di Mobile Programming - A.A. 2024/25

Studenti del gruppo 16:

Mattia Sanzari - Matricola 0612707696

Valerio Volzone - Matricola: 0612707693

Vittorio Postiglione - Matricola: 0612707620

Sharon Schiavano - Matricola: 0612708676

Angela Monti - Matricola: 0612708122

Docente:

Prof. Francesco Cauteruccio

Indice

Relazione di Progetto

1. Introduzione

Il presente documento descrive l'applicazione MyVinylCollection, progettata per la gestione digitale e l'organizzazione di collezioni di vinili. L'applicazione offre funzionalità complete per la catalogazione dettagliata, l'organizzazione strutturata e l'analisi statistica dei dischi posseduti. Permette agli utenti di registrare un'ampia gamma di dettagli specifici per ogni vinile, monitorarne lo stato di conservazione nel tempo, gestire e visualizzare le immagini di copertina, e consultare statistiche aggregate sulla propria collezione, offrendo dunque la possibilità di inventariare la propria collezione.

2. Requisiti Funzionali

I requisiti funzionali definiscono le capacità specifiche che il sistema deve possedere e le interazioni che l'applicazione deve supportare per soddisfare le esigenze degli utenti.

2.1. Gestione Generale della Collezione

- **Descrizione del Vinile:** Ogni elemento della collezione, ovvero ogni vinile, deve essere descritto tramite i seguenti attributi:
    - **Titolo:** Il titolo dell'album o del singolo.
    - **Artista:** Il nome dell'artista o del gruppo musicale.
    - **Anno di pubblicazione:** L'anno di rilascio del vinile.
    - **Genere musicale:** La categoria musicale a cui appartiene il vinile (es. Rock, Jazz, Classica).
    - **Etichetta discografica:** Il nome dell'etichetta che ha prodotto il vinile.
    - **Condizione:** Una classificazione dello stato fisico del vinile (es. "nuovo" per un disco sigillato, "usato" per un disco con segni di usura normali, "da restaurare" per un disco che necessita di interventi).
    - **Immagini di copertina:** Possibilità di associare e visualizzare immagini della copertina del vinile.
    - **Flag "preferito":** Un indicatore che consente all'utente di marcare i dischi di particolare rilevanza, facilitandone il recupero e la visualizzazione.
- **Operazioni sui Vinili:** L'utente deve poter eseguire le seguenti operazioni per la gestione del ciclo di vita di un vinile all'interno della collezione:
    - **Creazione di un nuovo vinile:** Inserimento di tutti i dettagli pertinenti per un nuovo vinile nella collezione.
    - **Lettura/visualizzazione dei dettagli:** Accesso e consultazione di tutte le informazioni associate a un vinile specifico.
    - **Modifica di un vinile esistente:** Aggiornamento delle informazioni di un vinile già presente, come la condizione o l'aggiunta di nuove immagini.
    - **Rimozione di un vinile:** Eliminazione definitiva di un vinile dalla collezione.

2.2. Schermata Principale

La schermata principale funge da punto di accesso centrale e offre una panoramica rapida e funzionalità di navigazione.

- **Lista dei Vinili Recenti:** Visualizzazione di una lista dei vinili più recenti aggiunti alla collezione, ordinati cronologicamente per data di inserimento, per tenere traccia delle ultime acquisizioni.
- **Suggerimenti Casuali:** Selezione casuale e presentazione di 3-5 vinili dalla collezione, offrendo all'utente spunti per riscoprire i propri dischi.
- **Azione Rapida "Aggiungi Vinile":** Un pulsante di accesso immediato che indirizza l'utente direttamente alla procedura di aggiunta di un nuovo vinile.

2.3. Schermata Dettaglio Vinile

Questa schermata è dedicata alla visualizzazione approfondita delle informazioni di un singolo vinile.

- **Visualizzazione Campi Informativi:** Presentazione chiara e organizzata di tutti i campi informativi del vinile (titolo, artista, anno, genere, etichetta, immagini di copertina), consentendo una consultazione immediata di tutti i dettagli.

2.4. Schermata Aggiunta/Modifica Vinile

Questa schermata consente l'inserimento di nuovi dati o l'aggiornamento di quelli esistenti.

- **Form di Inserimento:** Un form interattivo con tutti i campi necessari per la descrizione di un vinile.
- **Selettore Condizione:** Un componente per selezionare la condizione d'uso del vinile da un elenco predefinito.
- **Upload Immagini:** Funzionalità per il caricamento di una o più immagini di copertina dal dispositivo dell'utente.
- **Validazione Dati:** Implementazione di regole di validazione in tempo reale e al momento del salvataggio per garantire l'integrità e la correttezza dei dati (es. verifica che l'anno sia numerico e che i campi obbligatori non siano vuoti).

2.5. Schermata Categorie

Questa sezione permette la gestione e la consultazione dei generi musicali presenti nella collezione.

- **Elenco Generi:** Visualizzazione di un elenco completo di tutti i generi musicali identificati all'interno della collezione.
- **Conteggio Vinili per Genere:** Indicazione numerica del numero di vinili associati a ciascun genere.
- **Gestione Categorie Personalizzate:**

    - Pulsante "Nuova categoria" per la creazione dinamica di generi musicali personalizzati dall'utente.
    - Possibilità di rinominare o eliminare le categorie create dall'utente, offrendo flessibilità nella classificazione.

2.6. Schermata Ricerca e Filtri

Questa schermata è dedicata alla scoperta e al recupero di vinili specifici all'interno della collezione.

- **Campo di Ricerca Testuale:** Un campo di testo che consente la ricerca libera di vinili per titolo, artista o etichetta discografica, supportando la ricerca per parole chiave.
- **Filtri Selezionabili:** Opzioni di filtraggio per personalizzare i risultati della ricerca:
    - **Intervallo di anni:** Filtro per visualizzare i vinili pubblicati in un determinato periodo (es. dal 1970 al 1980).
    - **Selezione di uno o più generi:** Filtro per visualizzare i vinili appartenenti a specifici generi musicali.
    - **Condizione del vinile:** Filtro per visualizzare i vinili in una determinata condizione (es. solo "nuovi").
- **Risultati in Lista:** Visualizzazione dei risultati della ricerca e dei filtri in un formato lista compatto, mostrando la copertina, il titolo e l'artista per una rapida identificazione.

2.7. Schermata Analisi (Statistiche)

Questa schermata offre una rappresentazione visiva e quantitativa della collezione.

- **Numero Totale di Vinili:** Un indicatore numerico chiaro del numero complessivo di vinili presenti nella collezione.
- **Distribuzione per Genere:** Un grafico a torta che illustra visivamente la ripartizione percentuale dei vinili per genere musicale, fornendo una panoramica dei generi predominanti.
- **Andamento Crescita Collezione:** Un grafico a linee o a barre che illustra l'andamento cronologico dell'aggiunta di vinili alla collezione nel tempo, permettendo di osservare i periodi di maggiore crescita.
- **Elenco Vinili Più Vecchi:** Una lista dei vinili con l'anno di pubblicazione più antico.

3. Requisiti Non Funzionali (Attributi di Qualità)

I requisiti non funzionali definiscono le qualità intrinseche del sistema e i vincoli operativi che devono essere rispettati.

- **Manutenibilità:** Il codice sorgente dell'applicazione è stato progettato per essere modulare, ben documentato tramite commenti esplicativi e aderire a standard di programmazione consolidati. Questa strutturazione facilita la comprensione del codice da parte di terzi, semplifica l'individuazione e la correzione di difetti, e consente una facile estensione o modifica delle funzionalità in future implementazioni del progetto.
- **Scalabilità:** L'applicazione è stata concepita con l'obiettivo di supportare un'espansione della collezione di vinili senza che le prestazioni complessive subiscano degradazioni significative. L'utilizzo di SQLite per la persistenza dei dati contribuisce in modo sostanziale a questo obiettivo, garantendo che l'applicazione rimanga reattiva anche con un elevato volume di dati.
- **Usabilità:** L'applicazione è stata sviluppata con un focus sull'interfaccia utente. Si è puntato a offrire un'interfaccia intuitiva, coerente e di facile navigazione, che permetta agli utenti di accedere rapidamente a tutte le funzionalità e di completare le operazioni desiderate con il minimo sforzo cognitivo.
- **Performance:** L'applicazione è ottimizzata per garantire tempi di risposta rapidi per tutte le operazioni principali, quali il caricamento delle liste di vinili, l'esecuzione di ricerche e filtri complessi, e la visualizzazione dei dettagli di un singolo disco.

4. Scelte Implementative

Per la realizzazione dell'applicazione sono state adottate specifiche scelte architetturali e pattern di progettazione, unitamente all'integrazione di librerie esterne, al fine di garantire la robustezza, la manutenibilità, la scalabilità e le performance del sistema.

4.1. Architettura e Pattern di Progettazione

L'architettura dell'applicazione è stata definita seguendo principi di buona progettazione software, con l'obiettivo primario di separare le responsabilità dei diversi moduli e facilitare lo sviluppo, il testing e la manutenzione del software.

Singleton Pattern

- **Scopo:** Assicurare che esista un'unica istanza del servizio di gestione del database per l'intera durata dell'applicazione.
- **Implementazione:** Si è adottato un costruttore privato (`_internal()`) per impedire l'istanziazione diretta e un'istanza statica condivisa (`_instance`). L'accesso globale all'unica istanza avviene tramite un factory constructor, che garantisce che ogni richiesta di accesso al servizio di database restituisca sempre la medesima istanza già creata.
- **Vantaggi:** Previene efficacemente la creazione di connessioni multiple e ridondanti al database, centralizza la gestione dello stato della connessione e delle operazioni sul database.

Repository Pattern

- **Scopo:** Separare la logica di accesso ai dati dall'interfaccia utente.
- **Implementazione:** Un servizio dedicato, denominato `DatabaseService`, è stato incaricato di aggregare tutte le operazioni CRUD (Create, Read, Update, Delete) relative ai vinili e alle loro proprietà. Ogni metodo all'interno di questo servizio incapsula una specifica operazione di persistenza (es. `insertVinyl` per l'inserimento, `getAllVinyls` per il recupero).
- **Vantaggi:** Isola la logica di persistenza dei dati, rendendo il codice più modulare e testabile. Facilita inoltre una potenziale sostituzione del backend di persistenza con modifiche minime o nulle alla logica di business e all'interfaccia utente.

Lazy Initialization

- **Scopo:** Ottimizzare le prestazioni dell'applicazione ritardando l'inizializzazione della connessione al database fino al momento in cui non è effettivamente necessaria.
- **Implementazione:** Il database viene inizializzato e aperto solo al primo accesso tramite un `getter` dedicato (`database`).
- **Vantaggi:** Riduce significativamente il tempo di avvio dell'applicazione, migliorando la percezione di reattività da parte dell'utente, e consente un risparmio di risorse nel caso in cui il database non venga utilizzato immediatamente o affatto durante una sessione.

Transaction Pattern

- **Scopo:** Garantire l'atomicità delle operazioni complesse che coinvolgono modifiche al database.
- **Implementazione:** Vengono utilizzate transazioni esplicite (tramite il metodo `db.transaction()`) per operazioni che coinvolgono la manipolazione di dati su

più tabelle, come l'inserimento di un vinile che potrebbe implicare anche l'inserimento delle canzoni associate o l'aggiornamento di più campi correlati.
- **Vantaggi:** Previene la corruzione dei dati e mantiene l'integrità referenziale all'interno del database (es. tra vinili e canzoni tramite chiavi esterne), garantendo che il database rimanga in uno stato consistente anche in presenza di errori o interruzioni.

Dual Counting Strategy

- **Scopo:** Bilanciare la necessità di performance immediate per l'aggiornamento dell'interfaccia utente con l'accuratezza e la consistenza dei conteggi per i report e le statistiche dettagliate.
- **Implementazione:** Viene mantenuta una cache interna (un contatore veloce come `vinylCount`) che viene aggiornata in tempo reale per fornire feedback immediato all'UI. Una "source of truth" viene ottenuta tramite query aggregate dirette sul database (`GROUP BY COUNT`) per generare dati accurati e consistenti, utilizzati specificamente per le schermate di analisi e report.
- **Vantaggi:** Offre una reattività immediata all'utente per le informazioni di base e i conteggi rapidi, migliorando l'esperienza utente, mentre garantisce l'accuratezza e l'affidabilità dei dati per le analisi.

Strategy Pattern per Cross-Platform

- **Scopo:** Supportare l'esecuzione dell'applicazione su diverse piattaforme (mobile, web, desktop) da un'unica codebase.
- **Implementazione:** Viene selezionato dinamicamente il `databaseFactory` appropriato in base alla piattaforma di esecuzione (`kIsWeb`). Questo permette di utilizzare implementazioni diverse per la creazione del database a seconda dell'ambiente.
- **Vantaggi:** Garantisce la piena compatibilità dell'applicazione con Flutter Web e desktop senza la necessità di mantenere codebase separate o di apportare modifiche significative al codice base, ottimizzando lo sviluppo multi-piattaforma e riducendo la complessità.

Observer Pattern (Implicito)

- **Scopo:** Consentire all'interfaccia utente di reagire e aggiornarsi automaticamente e in tempo reale in risposta a modifiche nei dati sottostanti.
- **Implementazione:** I metodi che interagiscono con il database restituiscono oggetti `Future`, per ottenere l'integrazione con `FutureBuilder`\`StreamBuilder`.
- **Vantaggi:** Assicura la reattività dell'interfaccia utente, fornendo un'esperienza utente fluida e sempre aggiornata.

4.2. API e Librerie Utilizzate

Per visualizzare i progressi e le statistiche della collezione è stato adottato fl_chart, un package open-source per Flutter disponibile su https://pub.dev/packages/fl_chart.

Motivazioni della Scelta:

- **Ampia Personalizzazione:** La libreria offre un elevato grado di configurabilità, permettendo di adattare colori, dimensioni, animazioni e interazioni dei grafici alle specifiche esigenze di design e usabilità. Consente un controllo dettagliato su sezioni, spaziature, etichette e stili dei grafici.
- **Varietà di Grafici:** Supporta diverse tipologie di grafici essenziali per le analisi richieste, inclusi `PieChart` (grafico a torta per distribuzioni), `LineChart` (grafico a linee per andamenti temporali) e `BarChart` (grafico a barre per confronti).
- **Documentazione e Tutorial:** La disponibilità di una documentazione ufficiale completa e ben strutturata, corredata da esempi di codice chiari, insieme a numerosi video didattici e articoli disponibili online, ha facilitato notevolmente l'apprendimento e l'integrazione della libreria.
- **Integrazione Semplice:** La sua API è intuitiva e si integra perfettamente con il sistema di widget di Flutter, garantendo un'implementazione nativa e una piena compatibilità con le versioni più recenti del framework, minimizzando la complessità di integrazione.

L'integrazione di `fl_chart` consente di creare dashboard interattive e altamente personalizzabili, offrendo un'esperienza utente ricca e dinamica per l'analisi approfondita della propria collezione di vinili.

4.3.Struttura del database

Il database dell'applicazione è stato progettato utilizzando SQLite, un sistema di gestione di database relazionali leggero e autonomo, ideale per applicazioni mobili e desktop. La struttura del database è composta da tre tabelle principali: vinili, categorie e songs, che permettono di archiviare in modo organizzato tutte le informazioni relative alla collezione.

Schema Tabelle

```
Tabella vinili
```

```sql
CREATE TABLE vinili (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  title TEXT NOT NULL,
  artist TEXT NOT NULL,
  year INTEGER NOT NULL,
  genre TEXT NOT NULL,
  label TEXT NOT NULL,
  condition TEXT NOT NULL,
  isFavorite INTEGER NOT NULL DEFAULT 0,
  imagePath TEXT,
  dateAdded TEXT NOT NULL,
  notes TEXT
);
```

```
Tabella categorie

CREATE TABLE categorie (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL UNIQUE,
  description TEXT,
  vinylCount INTEGER NOT NULL DEFAULT 0,
  isDefault INTEGER NOT NULL DEFAULT 0,
  dateCreated TEXT NOT NULL
);
```

```
Tabella songs

CREATE TABLE songs (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  vinylId INTEGER NOT NULL,
  titolo TEXT NOT NULL,
  artista TEXT NOT NULL,
  anno INTEGER NOT NULL,
  trackNumber INTEGER,
  duration TEXT,
  FOREIGN KEY (vinylId) REFERENCES vinili (id) ON DELETE CASCADE
);
```

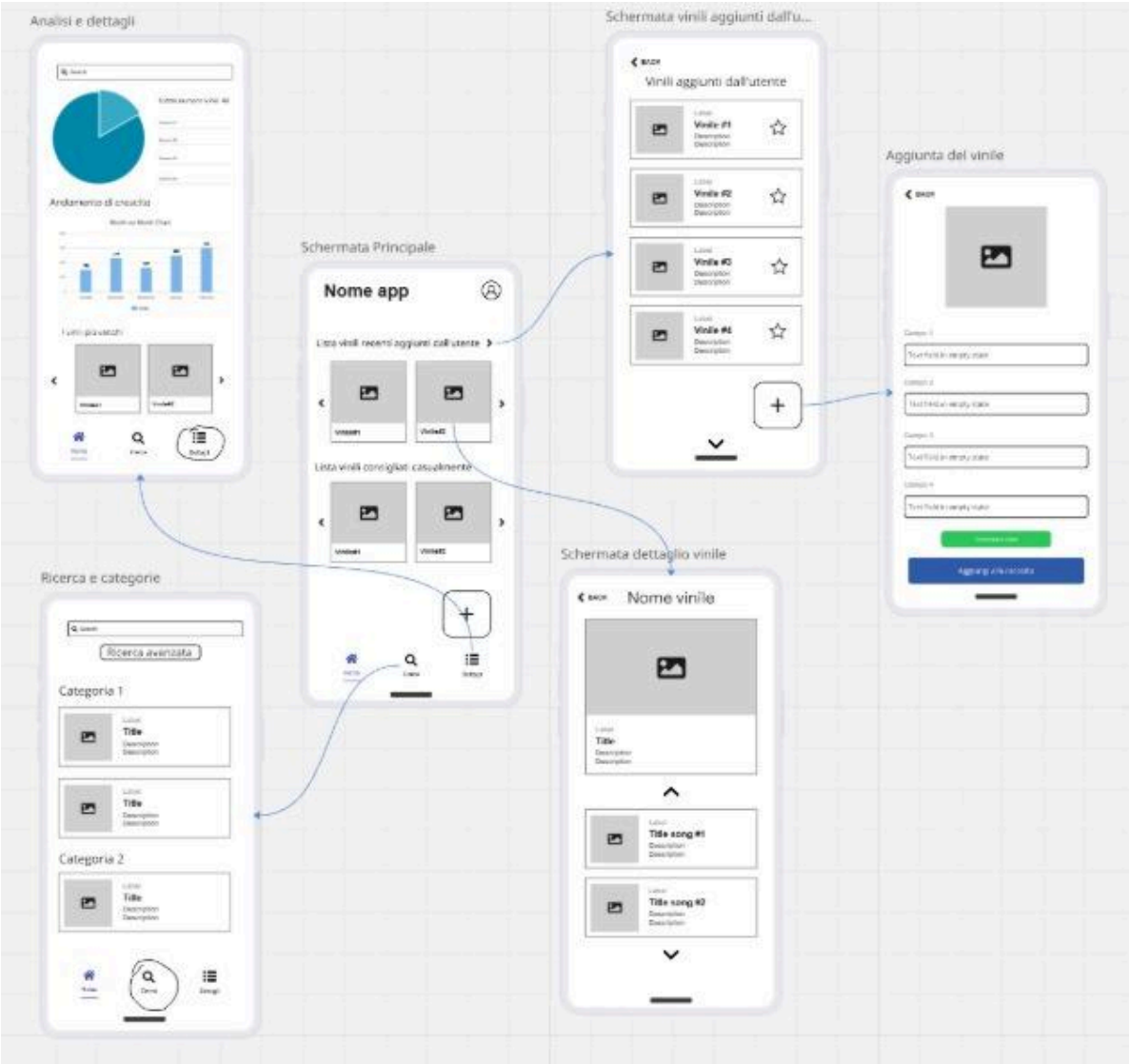
Relazioni tra le Tabelle

- **Vinyl ↔ Category:** Esiste una relazione uno a molti (1:N) tra la tabella categorie e la tabella vinili. Questo significa che un singolo genere (categoria) può essere associato a molti vinili, mentre ogni vinile appartiene a un solo genere.
- **Vinyl ↔ Song:** Esiste una relazione uno a molti (1:N) tra la tabella vinili e la tabella songs. Un singolo vinile può contenere molte canzoni, ma ogni canzone è associata a un solo vinile.

Integrità Referenziale

Per la relazione Vinyl ↔ Song, è stata implementata l'integrità referenziale tramite l'opzione ON DELETE CASCADE. Questo significa che se un record nella tabella vinili viene eliminato, tutte le canzoni correlate in songs (ovvero quelle con lo stesso vinylId) verranno automaticamente eliminate, garantendo la consistenza dei dati ed evitando record orfani.

4.4. Mockup



5. Conclusioni

L'applicazione MyVinylCollection rappresenta un'implementazione software completa per la gestione di collezioni di vinili. Il progetto ha dimostrato l'applicazione di principi di ingegneria del software, tra cui una architettura basata su pattern di progettazione consolidati e l'integrazione di librerie specializzate. Le scelte implementative adottate, unitamente a una struttura di database relazionale ben definita, garantiscono il rispetto dei requisiti non funzionali di manutenibilità, scalabilità e performance. Questo lavoro evidenzia la capacità di tradurre requisiti funzionali e non funzionali in una soluzione software strutturata e performante.