Progetto Mobile Programming A.A. 2024/25

Relazione Travel Diary App

Introduzione

Travel Diary App è un'applicazione mobile multipiattaforma sviluppata con **Flutter**, pensata per facilitare la registrazione, l'organizzazione e la visualizzazione delle esperienze di viaggio personali.
Il progetto è stato realizzato nell'ambito del corso di Mobile Programming, con l'obiettivo di applicare i diversi concetti appresi durante il percorso didattico, focalizzandosi sulla persistenza dei dati, la modularità del codice e un'interfaccia utente semplice e intuitiva.
L'app consente all'utente di creare schede dettagliate per ogni viaggio, associare immagini, marcare destinazioni preferite o da ripetere, e visualizzare statistiche aggregate sulle proprie avventure.

Funzionalità principali:

L'app è composta da una serie di schermate user-friendly, che guidano l'utente attraverso un processo intuitivo:

- **Creazione dei viaggi**: L'utente può creare nuove schede di viaggio specificando campi come titolo, località, date di inizio e fine, categoria (es. Cultura, Natura), note personali ed eventualmente associare immagini.
  Ogni viaggio può essere associato a un continente, una nazione o una città.
  Successivamente c'è la possibilità di associare una categoria personalizzata, fornendo flessibilità nella classificazione.

- **Visualizzazione dei viaggi**:
  - La **schermata principale (Home Screen)** offre una panoramica degli ultimi viaggi aggiunti, con immagini di copertina e switch visivi per indicare lo stato di "preferito" o "da ripetere".
  - Una sezione dedicata mostra le "Destinazioni preferite" in una galleria, permettendo un accesso rapido ai dettagli.

- **Gestione dettagliata dei viaggi**: Tramite la schermata di dettaglio, l'utente può:
  - Visualizzare tutte le informazioni complete del viaggio e sfogliare una galleria delle immagini associate.
  - Marcarlo come "preferito" o "da ripetere" tramite un bottone intuitivo.
  - Accedere alla modalità di modifica per aggiornare qualsiasi dettaglio del viaggio.
  - Eliminare definitivamente il viaggio dalla propria collezione.

- **Esplorazione e analisi dei viaggi**: L'app offre diverse modalità per esplorare e comprendere le proprie esperienze di viaggio:
  - **Schermata Categorie**: Visualizza i viaggi raggruppati per categoria, mostrando il conteggio per ciascuna e permettendo di filtrare rapidamente i viaggi di una specifica tipologia.
  - **Schermata Ricerca**: Consente di cercare e filtrare i viaggi per località, titolo, categoria o intervallo di date, fornendo uno strumento potente per trovare esperienze passate.

- **Schermata Analisi**: Presenta statistiche aggregate, come il numero totale di viaggi effettuati, la distribuzione per tipologia di viaggio, il numero medio di viaggi per anno e l'identificazione del paese/località più frequentemente visitato.

Il progetto è stato strutturato seguendo principi di modularità e separazione delle responsabilità, organizzato nelle seguenti macro-aree:

- **`lib/models`**: Contiene le classi che definiscono la struttura dei dati dell'applicazione.
- **`lib/helpers`**: Include le classi di utilità, in particolare quelle per la gestione del database.
- **`lib/screens`**: Raggruppa tutte le schermate dell'interfaccia utente dell'applicazione.
- **`lib/utils`**: Contiene dati statici e funzioni di utilità generali (es. `AppData` per le immagini di default).

Il file `main.dart` rappresenta il punto di ingresso dell'applicazione, configurando il tema globale e lanciando la schermata iniziale.

Classi Oggetti (Model)

La cartella lib/models contiene la classe Trip, che è il modello fondamentale per la rappresentazione di un viaggio.
Trip **(**lib/models/trip.dart**)**
La classe Trip definisce la struttura dei dati per ogni viaggio registrato nell'applicazione. Un oggetto Trip è caratterizzato da un identificativo univoco, un titolo, la località, le date di inizio e fine, note personali, una categoria, il continente, e due flag booleani per indicare se il viaggio è tra i preferiti o se è da ripetere. Include inoltre una lista di percorsi per le immagini associate al viaggio.
La classe fornisce i metodi necessari per la conversione dell'oggetto da e verso il formato utilizzato per la persistenza nel database.

Classi Interfaccia Grafica (Screens)

La cartella `lib/screens` contiene tutti i widget che rappresentano le diverse schermate dell'applicazione. Ogni schermata è un widget Flutter che gestisce la propria logica di presentazione e l'interazione con l'utente, mantenendo una chiara separazione delle responsabilità.

HomeScreen **(**lib/screens/home_screen.dart**)**
La schermata principale dell'app è stata progettata per fornire un accesso intuitivo alle funzionalità chiave della nostra applicazione.
Mostra un elenco degli ultimi viaggi aggiunti, con immagini di copertina e indicatori di stato (preferito/da ripetere), e una sezione dedicata alle destinazioni preferite organizzata in una galleria orizzontale.
Permette un accesso rapido alle altre sezioni dell'app e all'aggiunta di un nuovo viaggio.

AddEditTripScreen (lib/screens/add_edit_trip_screen.dart)
Questa schermata permette la creazione o la modifica di un viaggio. L'utente può inserire il titolo, la località, le date di inizio e fine, la categoria e note personali. Offre un elenco dinamico per la selezione del continente e della nazione, e permette di aggiungere e gestire più immagini dalla galleria del dispositivo, con anteprime e opzioni di rimozione. Sono presenti anche switch per marcare il viaggio come preferito o da ripetere.

TripDetailScreen (lib/screens/trip_detail_screen.dart)
Visualizza tutti i dettagli di un viaggio selezionato. La schermata presenta l'immagine di copertina del viaggio, il titolo, la località, le date, la categoria e le note personali. Sono inclusi indicatori visivi per lo stato di preferito o da ripetere. Una sezione dedicata mostra una galleria di tutte le immagini associate al viaggio.
L'utente può modificare o eliminare il viaggio direttamente da questa schermata.

CategoriesScreen (lib/screens/categories_screen.dart)
Fornisce l’insieme dei viaggi raggruppati per categoria. La schermata elenca tutte le categorie di viaggio esistenti, mostrando per ciascuna il numero di viaggi associati. Permette di navigare alla schermata di ricerca con un filtro pre-applicato per la categoria selezionata.

SearchScreen (lib/screens/search_screen.dart)
Permette agli utenti di cercare e filtrare i viaggi in base a vari criteri. Offre un campo di ricerca testuale per titolo o località e filtri per categoria e intervallo di date. I risultati della ricerca vengono visualizzati in un elenco con dettagli essenziali e immagini di copertina, e un pulsante permette di cancellare tutti i filtri applicati.

AnalysisScreen (lib/screens/analysis_screen.dart)
Offre statistiche aggregate sulle esperienze di viaggio dell'utente. La schermata calcola e visualizza dati come il numero totale di viaggi effettuati, la media di viaggi per anno, il conteggio dei viaggi preferiti e da ripetere, e la distribuzione dei viaggi per tipologia e per anno. Identifica anche il paese/località più frequentemente visitato.

Gestione Dati e Database

La persistenza dei dati è un pilastro fondamentale dell'applicazione, garantita dall'integrazione di un database SQLite locale.

Gestione Dati

L'applicazione gestisce lo stato principalmente a livello locale per ciascuna schermata. Le interazioni con il database sono mediate da una classe helper dedicata, che funge da repository per gli oggetti `Trip`. Questo approccio assicura che la logica di persistenza sia centralizzata e riutilizzabile.

Database

`DatabaseHelper` **(**`lib/helpers/trip_database_helper.dart`**)**
La classe `TripDatabaseHelper` è il cuore della persistenza dei dati. Implementa il pattern **Singleton**, assicurando una singola istanza della connessione al database in tutta l'applicazione. È responsabile dell'inizializzazione del database SQLite, della creazione dello schema delle tabelle e della gestione delle migrazioni per supportare l'evoluzione dei dati nel tempo. Fornisce inoltre tutte le operazioni necessarie per salvare, recuperare, aggiornare ed eliminare i viaggi dal database. Le funzioni implementate all'interno del Database helper vengono invocate nelle funzioni di aggiunta, modifica e cancellazione presenti nelle schermate dell'app.

Vincoli Tecnici

L'applicazione è stata sviluppata rispettando tutti i vincoli tecnici specificati:

- **Framework**: L'app è stata interamente sviluppata utilizzando **Flutter**, garantendo una singola codebase per Android e iOS.
- **Librerie e Package Esterni**: Sono stati ampiamente utilizzati pacchetti esterni per estendere le funzionalità di Flutter, come `sqflite` per il database, `image_picker` per la selezione delle immagini, `path_provider` per la gestione dei file, `intl` per la formattazione delle date e `cached_network_image` per la gestione efficiente delle immagini.
- **Interfaccia Responsive**: L'interfaccia utente è stata progettata per essere responsiva, adattandosi a diverse dimensioni di schermo e orientamenti.
- **Persistenza dei Dati**: I dati dell'applicazione sono **persistenti**, garantendo che tutte le informazioni sui viaggi vengano salvate sul dispositivo e siano disponibili anche dopo la chiusura e la riapertura dell'app.

- **SQLite**: È stato utilizzato **SQLite** per la persistenza dei dati, come consigliato dalla traccia, implementando un database robusto e ben strutturato per la gestione dei viaggi.

Conclusione

Il progetto "Travel Diary App" rappresenta un'applicazione completa e funzionale per la gestione dei diari di viaggio, progettata con un'interfaccia intuitiva e una struttura solida. Attraverso l'utilizzo del framework Flutter e l'integrazione di un database SQLite, è stato possibile realizzare un'app capace di offrire un'esperienza utente fluida e organizzata, mantenendo i dati persistenti e accessibili.

Durante lo sviluppo sono stati applicati e consolidati concetti fondamentali della programmazione mobile, come la chiara separazione tra logica e interfaccia (Model-View), la gestione della persistenza dei dati (SQLite con migrazioni), la modularizzazione del codice e l'attenzione alla User Experience. Il progetto ha offerto l'opportunità di approfondire le competenze tecniche e progettuali, ponendo le basi per eventuali estensioni future, come l'integrazione di servizi cloud o funzionalità di social sharing.