Relazione Grocery Tracking App

Introduzione

Gestore Spese è un'applicazione mobile sviluppata con Flutter, pensata per facilitare la gestione delle spese 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.
L'app consente all'utente di creare e organizzare liste della spesa, aggiungere prodotti, assegnarli a categorie specifiche e monitorare i costi in modo strutturato.
Attraverso un'interfaccia semplice e intuitiva, l'app punta a migliorare l'esperienza dell'utente nella pianificazione delle spese quotidiane.

Funzionalità principali:

Abbiamo strutturato il progetto secondo il pattern View-Model:

- La cartella Model contiene tutte le classi logiche e di struttura dati(Spesa, Prodotto, Categoria, e la gestione del database)
- La cartella View contiene tutte le schermate dell'interfaccia utente, ciascuna per una specifica funzionalità
- Il file main.dart rappresenta il punto di ingresso dell'app.

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

- **Creazione dei prodotti**, specificando campi predefiniti come nome, prezzo, quantità, ed eventualmente una descrizione. Ogni prodotto può essere associato a una categoria personalizzata.

- **Creazione delle liste della spesa**, selezionando i prodotti desiderati e specificando per ciascuno la quantità da acquistare.

- **Visualizzazione delle liste**, tramite una schermata dedicata che consente di accedere ai dettagli di ogni lista.

- **Gestione delle spese all'interno delle liste**, con la possibilità di:

    - Modificare i dettagli dei prodotti (come prezzo o note aggiuntive),

    - Segnare ciascuna voce come **acquistata** o **non acquistata**, facilitando la pianificazione e l'utilizzo della lista anche in date future.

Entriamo nel dettaglio di ogni oggetto ed il proprio ruolo nell'applicazione:

Classi Oggetti

Categoria:

Rappresenta le categorie di Prodotti create dall'utente, caratterizzate dal nome della categoria e dal numero di spese che la posseggono;

Prodotto:

Rappresenta i prodotti che possono entrare a far parte di una Spesa e sono caratterizzate dal nome del prodotto, il suo prezzo, la Categoria a cui appartiene e le sue note (se l'utente vuole aggiungerne);

Spesa:

Rappresenta le spese effettuate o programmate dall'utente, registrate all'interno di una lista della spesa e caratterizzate dal Prodotto a cui fa riferimento, la data di creazione della Spesa, la quantità di oggetti acquistati del relativo Prodotto e un valore booleano per segnare la Spesa come "fatta" o "da fare";

ListaSpese:

Rappresenta le liste della spesa create dall'utente, caratterizzate dal nome della lista, il prezzo totale delle spese al suo interno, un oggetto List<Spesa> che rappresenta l'effettiva lista di oggetti Spesa, il numero di elementi al suo interno e la sua data di creazione.

Classi Interfaccia Grafica

main:

il file dove viene lanciata la home dell'app;

AddList (Widget Stateful):

Questa pagina permette la creazione vera e propria della lista; dopo aver selezionato il nome della lista da creare nel popup **InfoListaScreen**, in questa schermata è possibile selezionare i prodotti da inserire nella lista specificando anche la quantità. La pagina è strutturata in due componenti principali:

- Interfaccia: barra di ricerca, lista di **SpesaView** (componenti appositamente creati per la creazione di istanze Spesa a partire da Prodotto)
- Logica: Stringa query, mappa ListaSpesaMomentanea

La barra di ricerca è collegata con la stringa query: quando nella TextField della barra di ricerca viene scritto qualcosa (onChanged) viene cambiato lo stato del widget e ricreata la listaFiltrata che, inserita nel ListView.builder, consente di mostrare tutti i prodotti cercati.

La lista di SpesaView è collegata con la mappa ListaSpesaMomentanea: ogni cambiamento al prodotto contenuto nella SpesaView viene scritto nella mappa, salvando come chiave il prodotto e come valore la sua quantità; quando viene filtrata la lista in base al valore di ogni prodotto nel filtro viene ricostruito in modo coerente il widget.

BloccoLista:

Visualizza a schermo un elemento card che mostra le principali caratteristiche di una lista della spesa (nome, data di creazione e spesa totale), oltre ad un bottone per eliminare la lista (con richiesta di conferma per procedere) e un altro (**>**) per mostrare le Spese create all'interno di quella lista;

CreaCategoriaView:

(HomeView => bottone **+** => bottone Crea Prodotto => bottone Crea Categoria
OPPURE
HomeView => bottone **+** => bottone Crea Categoria)

Una semplice pagina per creare una nuova Categoria, dove si può inserire il nome della nuova categoria e confermare la creazione cliccando bottone Conferma oppure annullarla e tornare indietro cliccando il bottone Annulla;

CreaProdottoView:

(HomeView => bottone **+** => bottone Crea Prodotto)

Una schermata che, dall'alto verso il basso, permette di inserire il nome del Prodotto da creare, selezionare una Categoria, creare una nuova Categoria cliccando il bottone Crea Categoria, inserire il prezzo unitario del Prodotto, inserire delle note e creare il Prodotto definito sopra cliccando il bottone Conferma o svuotare il form e tornare indietro cliccando il bottone Annulla;

HomeView:

Mostra i prodotti creati di recente tramite widget di tipo **ProdottoView**, la loro barra di ricerca per nome, una tab bar in fondo allo schermo per passare alla visualizzazione delle liste recenti e un bottone in basso a destra dello schermo per poter creare una nuova Categoria, Prodotto o Lista;

InfoListaScreen:

(HomeView => bottone **+** => bottone Crea Lista)

Una tendina che dal fondo alla metà dello schermo si apre, permettendo di inserire il nome della lista da creare e un bottone Avanti cliccabile solo se la Lista ha un nome inserito;

ListaView:

Mostra l'interno di una lista della spesa: nell'appBar viene visualizzato il nome della Lista; nella pagina viene visualizzata una barra di ricerca delle spese, la data di creazione della lista, il prezzo totale delle spese contenute al suo interno e un loro elenco
(per ogni Spesa, mostra una card che rappresenta le principali informazioni di una Spesa all'interno di una lista (nome, prezzo unitario, quantità di oggetti dello stesso Prodotto appartenenti alla Spesa e il costo totale della Spesa), un bottone (**>**) per mostrare tutti i dettagli della Spesa, un bottone a forma di matita per modificare la Spesa e un checkbox spuntabile che segna se la Spesa è stata fatta o meno);

ListeView:

Mostra le liste create di recente, una tab bar in fondo allo schermo per passare alla visualizzazione dei prodotti recenti e un bottone in basso a destra dello schermo per poter creare una nuova Lista;

ProdottoView:

Mostra una card contenente le principali informazioni di un Prodotto (nome, categoria e prezzo);

SpesaProdottoView

(ListaView => CustomCards => bottone **>**)

Mostra tutti i dettagli di una Spesa (nome del Prodotto a cui fa riferimento, il prezzo unitario, le note e la quantità di oggetti di quel Prodotto appartenenti alla Spesa);

SpesaView (Widget Stateful):

Questa classe rappresenta il blocco visibile nella schermata AddList. E' composto da quattro elementi fondamentali: selected (booleano), count (intero), Prodotto e targetMap (la mappa passatagli dall'interfaccia AddList); inoltre nello stato viene dichiarato un oggetto spesa da gestire con i pulsanti. Il funzionamento dei pulsanti è il seguente:

- Utente spunta la checkbox => viene automaticamente incrementato di 1 il count e viene salvato l'oggetto nella mappa (per permettere al widget AddList di gestire la creazione di questo widget ogni volta che cambia il testo nella barra di ricerca) e viene salvato nella lista spese del Provider GestoreApp (consentendo così di ospitare temporaneamente in una lista accessibile alle schermate gli oggetti Spesa creati dai Prodotti in SpesaView)

- Utente incrementa o diminuisce il contatore => viene aggiornato il valore della chiave nella mappa e aggiornata l'istanza Spesa dichiarata nello stato (ad ogni cambiamento della quantità viene reinserita l'istanza di Spesa nella lista spese del GestoreApp);

  Nota: se il contatore va a zero viene deselezionato il Prodotto e rimosso sia dalla mappa che dalla lista spese.

Modello Dati

GestoreApp:

La gestione dei dati viene fatta in questo file dart implementando il modello del Provider. Ogni cambiamento degli attributi nella classe verrà notificato a tutte le schermate dell'app. Gli attributi sono delle liste che contengono: tutti i prodotti che l'utente ha inserito, tutte le categorie create, tutte le liste della spesa, una lista di appoggio usata nella schermata di creazione della lista che ospita (temporaneamente) le spese da inserire. Abbiamo optato per questo modello di dati per avere una classe centralizzata visibile in tutti i punti dell'applicazione come ruolo di contenitore di tutti i dati visibili a schermo, su cui è possibile implementare tutte le funzionalità dell'app.

Database

DatabaseHelper:

Abbiamo implementato un database sqlite per permettere il salvataggio dei dati. Questa classe si occupa della gestione di creazione del database su macchina locale e gestione delle tabelle associate alle classi oggetto dell'app. Le funzioni implementate all'interno del Database helper vengono invocate in ogni funzione di aggiunta, modifica e cancellazione presente nel **GestoreApp** (verificabile nel codice sorgente).

Conclusione

Il progetto Gestore Spese rappresenta un’app completa e funzionale per la gestione delle spese quotidiane, progettata con un’interfaccia intuitiva ed una struttura solida. Attraverso l’utilizzo dei framework Flutter e l’integrazione di un database, è stato possibile realizzare un’app capace di offrire un’esperienza utente fluida ed organizzata.

Durante lo sviluppo sono stati applicati concetti fondamentali della programmazione mobile, come la separazione tra logica ed interfaccia, la gestione della persistenza dei dati e la modularizzazione del codice. Il progetto ha offerto l’opportunità di consolidare le competenze tecniche e progettuali, ponendo le basi per eventuali estensioni future.

Gruppo 15, Mobile Programming a/a 2024/2025
Rumma Giovanni 0612707353
Alessandro Senatore 0612708130
Vincenzo Nazzaro 0612708341
Carmine Mirante 0612708187
Antonio Pacilio 0612707605

.