Università degli Studi di Salerno

.DIEM

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Corso di Laurea in Ingegneria Informatica

Mobile Programming

Progetto 4 – Gestione eventi

Gruppo 30

Studenti:
Francesco Pio Cirillo mat.0612705370
Alessandra Fasolino mat.0612705401

Anno accademico 2023-2024

Sommario

1. Descrizione tecnica

1.1. Framework e struttura

Il progetto è stato realizzato con il framework Flutter. Il Widget più importante è sicuramente HomePage che ospita le tre schermate fondamentali dell'app (Dashboard, Management e Statistiche) divise e accessibili tramite una NavigationBar.

Per mezzo di una classica navigazione stack è possibile accedere alla schermata NewEvent tramite la pressione del FloatingActionButton apposito; da questa pagina si ritorna alla precedente con l'apposito tasto indietro nella AppBar o a seguito della conferma dell'aggiunta del nuovo evento sempre attraverso un FloatingActionButton.

Sempre per mezzo di una classica navigazione stack è possibile accedere ad una schermata di dettaglio su un singolo evento per mezzo della pressione su di esso.

Si è prestato attenzione all'utilizzo di SafeArea in tutte le schermate al fine di salvaguardare il corretto rendering dei Widget sullo schermo in armonia con la status bar.

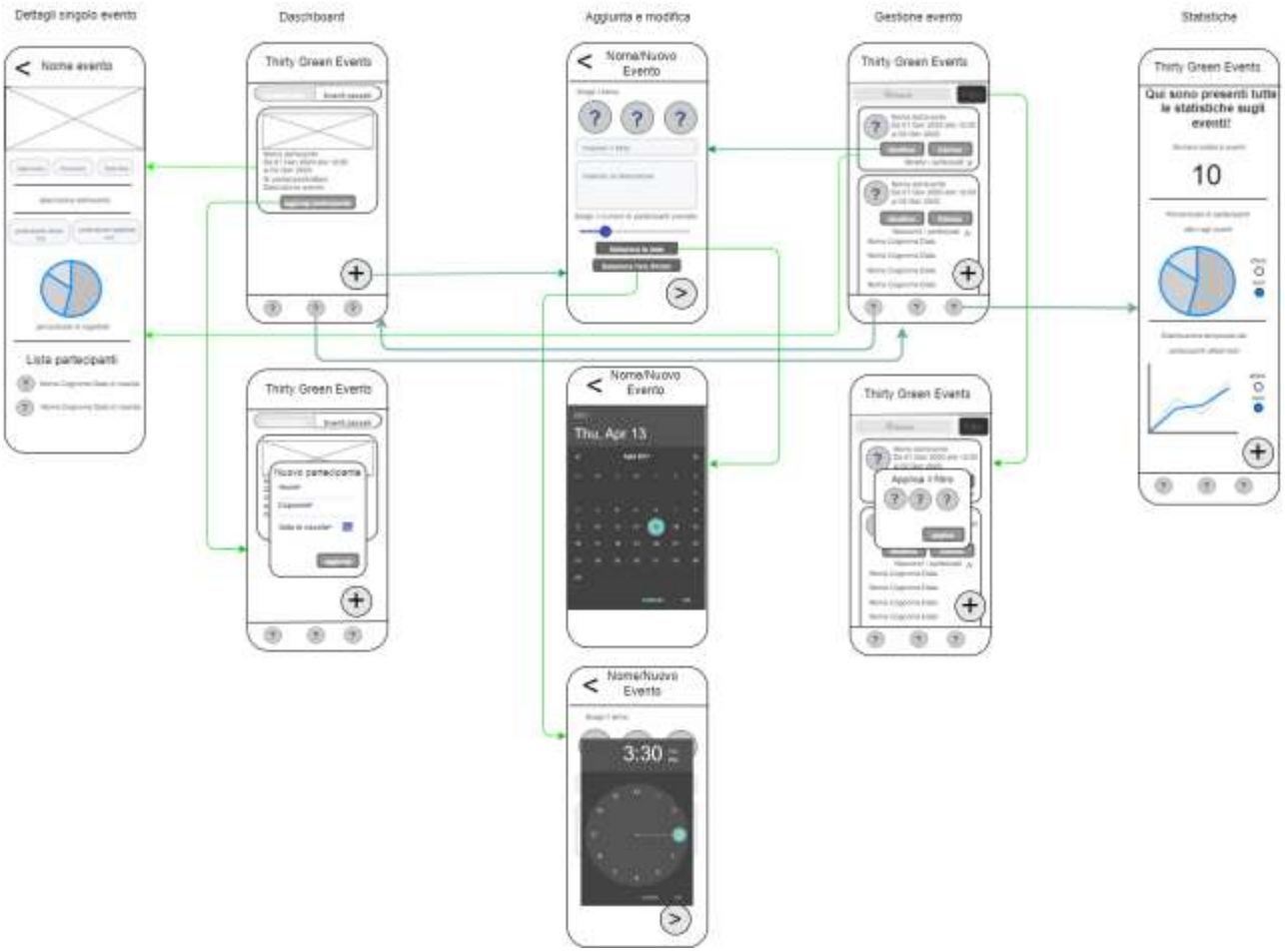

Figura 1 - Wireframe

1.2. Persistenza dei dati

La persistenza dei dati è stata realizzata per mezzo dell'utilizzo di SQLite in modo da permettere potenzialmente la memorizzazione di una mole di dati significativa.

Parte dello script di creazione del DataBase aggiunge anche in automatico una serie di Eventi con annessi partecipanti a scopo illustrativo.

1.3. Stile e tema

Il tema dell'applicazione è stato definito per mezzo del parametro Theme del widget MaterialApp nel main, che è radice dell'albero di Widget dell'applicazione sviluppata.

2. Overview

Di seguito sono descritte le caratteristiche principali delle schermate dell'app e di altri elementi rilevanti.
Il nome dell'applicazione è 30 green events per dar risalto al numero identificativo del gruppo e al tema della traccia che tratta di gestione di eventi, in accordo con lo stile dell'applicazione si è anche creato il logo.

Figura 2 - logo

2.1. Cinque Schermate

Figura 3- Schermata 1 Dashboard

Figura 3.1 - Form aggiunta partecipante

La Dashboard permette di visualizzare tutti gli eventi divisi in eventi passati ed eventi futuri tramite dei ToggleButtons. La Card di ogni evento permette di aggiungere un nuovo partecipante attraverso la pressione di un ElevatedButton specificando poi le sue informazioni in un AlertDialog.

Cliccando sulla Card di un evento è possibile, come anche dalla schermata 2, accedere alla pagina relativa ai dettagli di un singolo evento, ciò è ottenuto attraverso la navigazione stack.

Cliccando sull'apposito FloatingActionButton è possibile, come anche dalle schermate 2 e 4, accedere alla schermata 3 per la creazione di un nuovo evento.

Figura 4- Schermata 2 Gestione evento

Figura 4.1 – AlertDialog per il filtro

Figura 4.2 – ExpansionPanel aperto

In cima alla Schermata gestione evento figurano in una Row una SearchBar e un ElevatedButton che permettono rispettivamente di cercare eventi per titolo e filtrare gli eventi sulla base del tema, selezionabile in un apposito AlertDialog.

Su ogni singola Card sono presenti due TextButton che permettono di modificare l'evento, spostandosi nella schermata 3 appositamente compilata, oppure di eliminare l'evento.

Nella parte inferiore di ogni Card è presente un ExpansionPanel che permette di visualizzare la ListView dei partecipanti registrati all'evento. Si può notare che l'elenco puntato dei partecipanti è decorato in base al tema dell'evento selezionato.

Figura 5- Schermata 3 Aggiunta e modifica (mod aggiunta)

Figura 5.1 - Schermata 3 Aggiunta e modifica (mod modifica)

In cima alla Schermata Aggiunta e modifica una Row contiene le tre possibili scelte per il tema dell'evento: lavoro, cibo, romantico. Per permettere la scelta sono stati usati dei CircleAvatar per presentare le immagini, i CircleAvatar sono posti all'interno di Container il cui borderColor diventa rosso se vengono selezionati.

Sono poi presenti due TextFormField che permettono di inserire il titolo e la descrizione dell'evento, a seguire uno Slider per indicare il numero di partecipanti atteso e in fine due ElevatedButton per selezionare, rispettivamente, il DateTimeRange quindi data di inizio e fine dell'evento e il TimeOfDay cioè l'ora d'inizio dell'evento.

Tutte le informazioni sono poi passate tramite una pop() sul Navigator alla pagina precedente che si occupa di aggiungere l'evento creato (o aggiornare l'evento modificato). Durante l'aggiornamento dei dati compare uno ScaffoldMessenger per mostrare un messaggio di caricamento.

Figura 6- Schermata 4 Statistiche

La Schermata Statistiche mostra 3 differenti statistiche di cui due su grafici della libreria fl_chart:

- Numero di eventi salvati sull'applicazione;
- PieChart che mostra la percentuale di partecipazione effettiva rispetto a quella attesa;
- LineChart che mostra la partecipazione attesa rispetto a quella effettiva durante lo scorrere di mesi.

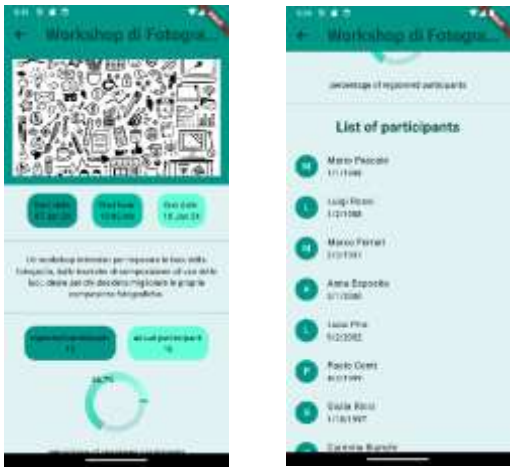

Figura 7- Schermata 5 Dettagli singolo evento

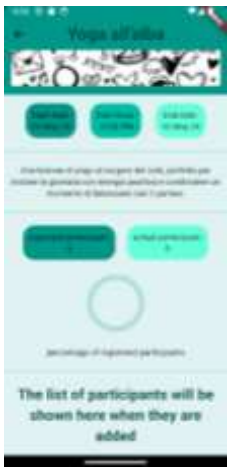

Figura 7.1 – Evento senza partecipanti registrati

La Schermata Dettagli singolo evento, non richiesta esplicitamente dalla traccia, è stata ideata per fornire informazioni dettagliate e ben organizzate rispetto ad un singolo evento di interesse.

La schermata in questione mostra, appena sotto all'immagine di asset corrispondete al tema dell'evento, una Row contenente tre BoxDecoration per mostrare le date di inizio e fine evento e l'orario d'inizio dell'evento.

Scorrendo in basso viene mostrata la descrizione dell'evento e successivamente un PieChart che rappresenta la percentuale di capienza dell'evento.

Infine è disponibile la ListView di persone che sono state registrate oppure un messaggio se l'evento ancora non è stato popolato di alcun partecipante.

2.2. Navigazione

Sono state utilizzate due forme di navigazione:

- la classica navigazione a pila, vista a lezione, per accedere alla schermata 3 cliccando il FloatingActionButton (Figura 5) delle schermate 1, 2 e 4 e per accedere alla schermata 5 cliccando sulle Cards (Figura 6) delle schermate 1 e 2;

Figura 8 - FloatingActionButton

Figura 8.1 - Card

- la navigazione a tab, tramite NavigationBar (Figura 7) approfondita tramite la seguente pagina della documentazione ufficiale di Flutter: https://api.flutter.dev/flutter/material/NavigationBar-class.html . La NavigationBar permette di renderizzare un widget diverso da un array di widget a seconda del tab selezionato nella barra di navigazione posizionata in basso alla pagina.

Figura 9 - NavigationBar

2.3. Form input

Sono state usate due modalità per gestire i campi di input:

- Tramite un Form e dei TextFormField nella schermata 3, gli input e i relativi messaggi di errore sono stati controllati singolarmente usando il validator del TextFormField.

Figura 10- Messaggi d'errore schermata 3

- Tramite un controllo nella funzione submitAddPerson e dei TextField per assicurarsi che i campi siano tutti compilati, renderizzando un messaggio di errore se questa condizione non è rispettata.

Figura 11 - Messaggio d'errore schermata 1

Per rendere l'app più interattiva è stata aggiunta una SnackBar temporanea gestita tramite il widget ScaffoldMessanger che mostra un messaggio di attesa durante il caricamento delle informazioni da salvare dopo aver confermato i form.

Figura 12 - SnackBar

2.4. Responsività

2.4.1. Diverse dimensioni di schermo

Per rendere l'applicazione correttamente adattabile a diverse dimensioni di schermo, la fase di testing e implementazione è stata affiancata dall'utilizzo di due differenti dispositivi emulati:

- Pixel_3°_API_extension_level7_x86_64 di circa 5'',
- Start Pixel Tablet API 30 di circa 11''.

Sono state applicate dimensioni relative allo schermo a grafici e immagini ed è utilizzato il widget Expanded per far adattare in modo ottimale i vari widget ai cambiamenti di dimensione di schermo.

Dato che nelle foto illustrative della documentazione sono mostrati screen del primo device si riportano di seguito screen del secondo:

Figura 13- Schermata 1 su 11''

Figura 14- Schermata 2 su 11'

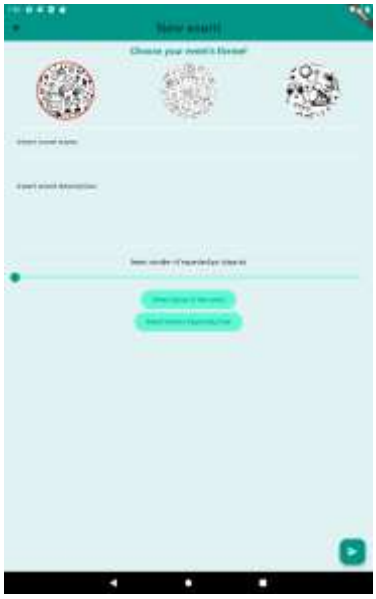

Figura 15- Schermata 3 su 11'

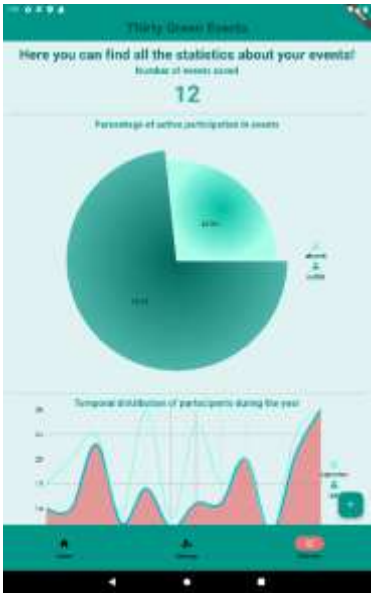

Figura 16- Schermata 4 su 11''

Figura 17- Schermata 5 su 11''

2.4.2. Portrait e landscape

Al fine di ottimizzare la user experience sono state prese scelte volte a migliorare l'usabilità dell'applicazione anche quando il dispositivo è in landscape mode, nello specifico sono state impostate diverse dimensioni di default per le immagini, per il LineChart e per la NavigationBar quando in modalità orizzontale.

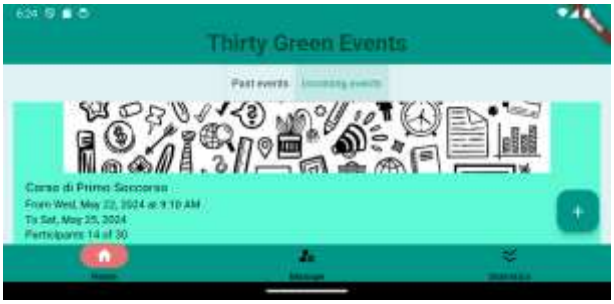

Figura 18 - Schermata 1 in landscape

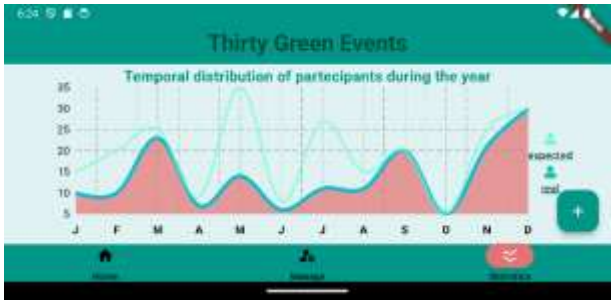

Figura 19 – Schermata 5 in landscape

3. Problematiche affrontate

3.1. ListView nella schermata 5

ListView di default occuperebbe tutto lo spazio verticale disponibile del widget che la ospita. Dato che è posizionata in una Column nel SingleChildScrollView, la ListView causava un errore di overflow.

Si è ritenuto quindi opportuno rendere la proprietà shrinkWrap vera in modo da poter gestire automaticamente la lunghezza della lista e renderla esattamente quella dei componenti che contiene.

Inoltre le ListView offrono già di default un meccanismo di scrolling quindi, se la lista era abbastanza lunga da ricoprire tutta la schermata, lo scrolling della lista stessa impediva il normale scorrimento della pagina impedendo all'utente di tornare in cima alla schermata.

Questo malfunzionamento è stato eliminato togliendo la possibilità di scrollare la lista tramite l'utilizzo della proprietà physics fornita dal widget stesso.

3.2. Dimensione dei grafici

Durante l'introduzione dei grafici è stato difficoltoso gestire l'overflow che si verificava al momento della loro renderizzazione. I grafici necessitano infatti di un container che limiti la loro dimensione, di conseguenza una soluzione come inserirli in un Expanded non è funzionale.

Si è quindi dimostrato necessario inserire ogni grafico in un SizedBox; inoltre per ottenere una dimensione adattiva e non fissa è stato necessario impostare width e height del SizedBox con valori derivati accedendo alle proprietà dell'oggetto MediaQuery.of(context).size.