

# PLANT CARE



Relazione progetto

Mobile Programming  
(A.A. 2024/25).

Gruppo 22

Antonio Della Porta,  
matricola: 0612710649

APPLICAZIONE  
MULTIPIATTAFORMA

## Sommario

Introduzione	2
Analisi del problema	2
Requisiti e funzionalità	3
Requisiti Funzionali	3
Requisiti Non Funzionali	3
Progettazione e tecnologie adottate	4
Progettazione grafica	4
Tecnologie utilizzate	4
Architettura software	4
Sviluppo delle funzionalità	4
Sviluppo	5
UI	5
Struttura delle cartelle e organizzazione del codice	9
Screens/mobile	10
Utils/chart	10
Utils/component	11
Descrizione del comportamento	11
Gestione dei dati	12
Statistiche e Visualizzazioni	14
Testing e Debugging	15
Conclusioni	16
Possibili sviluppi futuri	17
Difficoltà affrontate e soluzioni adottate	17
Allegati	17

## Introduzione

Relazione che documenta lo sviluppo di un'applicazione mobile realizzata come progetto per l'esame di **Mobile Programming (A.A. 2024/25)**.

L'obiettivo del progetto è quello di progettare e realizzare **un'app** multiplatforma per la **gestione e il monitoraggio della propria collezione di piante**, destinata a utenti appassionati di botanica, giardinaggio o semplicemente interessati alla cura delle proprie piante domestiche.

L'app consente agli utenti di:

- **Registrare le proprie piante**, inserendo informazioni dettagliate come nome, descrizione, categoria e data di inserimento.
- **Categorizzare le piante**, in modo da organizzare la collezione per tipologia o altre caratteristiche personalizzabili.
- **Gestire le attività di cura**, in particolare:
  - **Innaffiatura** (watering)
  - **Potatura** (pruning)
  - **Rinvaso** (transfer)

Per ogni pianta è possibile monitorare lo stato e le attività eseguite o da eseguire, garantendo così un'assistenza costante per la loro salute e crescita. Inoltre, l'app fornisce **statistiche e grafici** che aiutano a visualizzare le attività di manutenzione svolte e lo stato della propria collezione.

L'app è stata sviluppata con **Flutter**, framework scelto per la sua capacità di generare app native per dispositivi Android e iOS a partire da un unico codice sorgente. La realizzazione ha previsto non solo la progettazione della **User Interface (UI)**, ma anche l'organizzazione di una solida **architettura del codice** e la gestione di un **database locale** per il salvataggio persistente delle informazioni.

## Analisi del problema

Prendersi cura di una collezione di piante richiede attenzione costante e organizzazione, specialmente quando il numero di esemplari cresce o quando le esigenze di ciascuna pianta variano per specie, stagionalità o condizioni ambientali. Spesso gli utenti tendono a dimenticare attività fondamentali come l'innaffiatura, la potatura o il rinvaso, mettendo a rischio la salute delle piante.

Il problema nasce dalla **mancanza di uno strumento pratico e personalizzato** che permetta di:

- **Registrare e categorizzare le piante possedute** in modo semplice.
- **Monitorare e pianificare le attività di cura**, ricordando con precisione le scadenze di innaffiatura, potatura e rinvaso.
- **Analizzare lo stato generale della collezione** tramite statistiche o grafici intuitivi.

L'app sviluppata intende quindi rispondere a questa esigenza, proponendosi come uno **strumento di supporto per appassionati, hobbisti e coltivatori**, offrendo una soluzione digitale che possa sostituire metodi tradizionali (come appunti cartacei o promemoria generici sul telefono), spesso inadeguati e poco efficaci.

Il **target di riferimento** comprende:

- Appassionati di piante e giardinaggio domestico.
- Persone che vogliono mantenere in salute la propria collezione di piante ornamentali o da appartamento.
- Utenti che desiderano organizzare le cure delle piante in modo metodico e tracciabile.

L'app nasce per un utilizzo mobile, con successivo sviluppo multiplatforma, così da garantire agli utenti un accesso immediato alle informazioni e la possibilità di registrare attività in modo pratico da qualsiasi dispositivo portatile.

## Requisiti e funzionalità

Per garantire un'applicazione efficace, funzionale e usabile, sono stati definiti una serie di **requisiti funzionali e non funzionali** che hanno guidato lo sviluppo del progetto.

### Requisiti Funzionali

I requisiti funzionali rappresentano le funzionalità principali che l'app deve offrire agli utenti:

- **Registrazione di una pianta:** l'utente può aggiungere una nuova pianta indicando nome, descrizione, categoria di appartenenza e data di inserimento.
  - **Categorizzazione delle piante:** possibilità di associare ogni pianta a una categoria esistente o di crearne di nuove per una migliore organizzazione.
  - **Gestione attività di cura:** per ogni pianta è possibile registrare:
    - Stato di innaffiatura.
    - Stato di potatura.
    - Stato di rinvaso.
  - **Monitoraggio delle scadenze:** l'app fornisce un elenco delle piante con le attività di cura da eseguire, suddivise per tipologia di intervento.
  - **Visualizzazione delle statistiche:** grafici e statistiche che mostrano:
    - Numero totale di piante registrate.
    - Distribuzione per categoria.
    - Distribuzione mensile delle attività svolte sulle piante.
    - Stato generale della collezione.
  - **Modifica e rimozione:** possibilità di modificare o eliminare piante e categorie già inserite.
- 
- **Persistenza dei dati:** tutte le informazioni devono essere salvate localmente nel dispositivo tramite un database persistente ( SQLite).

### Requisiti Non Funzionali

I requisiti non funzionali definiscono le qualità che l'app deve rispettare:

- **Interfaccia utente semplice e intuitiva:** la UI deve essere facilmente navigabile anche da utenti senza esperienza tecnica.

- **Performance:** l'app deve garantire rapidità di caricamento e fluidità nell'interazione, anche con un numero elevato di piante registrate.
- **Scalabilità:** il codice deve essere strutturato per permettere facilmente l'aggiunta di nuove funzionalità, come ad esempio notifiche o promemoria.
- **Usabilità offline:** l'app deve funzionare completamente offline, senza necessità di connessione internet.
- **Accessibilità:** il design deve essere accessibile, con un'interfaccia chiara e leggibile.

## Progettazione e tecnologie adottate

La progettazione dell'app è stata guidata da principi di **modularità, chiarezza e separazione delle responsabilità**, con l'obiettivo di realizzare un'app solida, facilmente manutenibile e scalabile.

### Progettazione grafica

Per la progettazione grafica e il **design dell'interfaccia utente (UI)** è stato utilizzato **Figma**, un tool professionale per il design di interfacce digitali. Questo ha permesso di definire in anticipo la struttura delle schermate, la disposizione degli elementi e il flusso di navigazione dell'utente, assicurando un'esperienza coerente e intuitiva.

### Tecnologie utilizzate

- **Linguaggio e framework:** l'app è stata sviluppata utilizzando **Flutter**,
- **Database locale:** per la gestione e la persistenza dei dati è stato adottato **SQLite**, integrato nell'app attraverso un'apposita classe che centralizza le operazioni di lettura e scrittura.
- **Backup dei dati:** ad ogni modifica del database viene generato automaticamente un **file JSON di backup**, che rappresenta un'esportazione completa dei dati. Questo meccanismo garantisce la sicurezza delle informazioni e la possibilità di recuperare facilmente la collezione di piante in caso di necessità.
- **IDE:** lo sviluppo è stato condotto tramite **Visual Studio Code (VS Code)**.

### Architettura software

L'app segue un'architettura basata su una **separazione logica dei componenti**:

- La gestione dei dati (modello e database) è completamente isolata in un modulo dedicato.
- I widget UI si interfacciano con il database attraverso funzioni e metodi specifici, questo permette di aggiornare dinamicamente la UI al variare dei dati, garantendo un'app reattiva e performante.

Questa organizzazione rende l'app facilmente estendibile per l'integrazione futura di nuove funzionalità, o modifica dei componenti esistenti.

### Sviluppo delle funzionalità

L'app è stata strutturata in **quattro pagine principali**, ciascuna progettata per raggruppare in modo ordinato e funzionale le diverse operazioni che l'utente può svolgere.

- La **Home** rappresenta il punto di ingresso, offrendo una panoramica immediata delle informazioni essenziali e un accesso facilitato alle funzioni principali legate alla gestione delle piante.

- La sezione **Categorie** consente invece di organizzare le piante in gruppi personalizzabili, facilitando la classificazione e la consultazione della propria collezione.
- Nella sezione **Ricerca**, l'utente può facilmente trovare le piante registrate tramite strumenti di ricerca e filtri, pensati per agevolare la navigazione anche in collezioni ampie o variegate.
- Infine, la pagina dedicata alle **Statistiche** fornisce una visione complessiva e sintetica dei dati registrati, permettendo di monitorare l'andamento e la gestione complessiva delle piante.

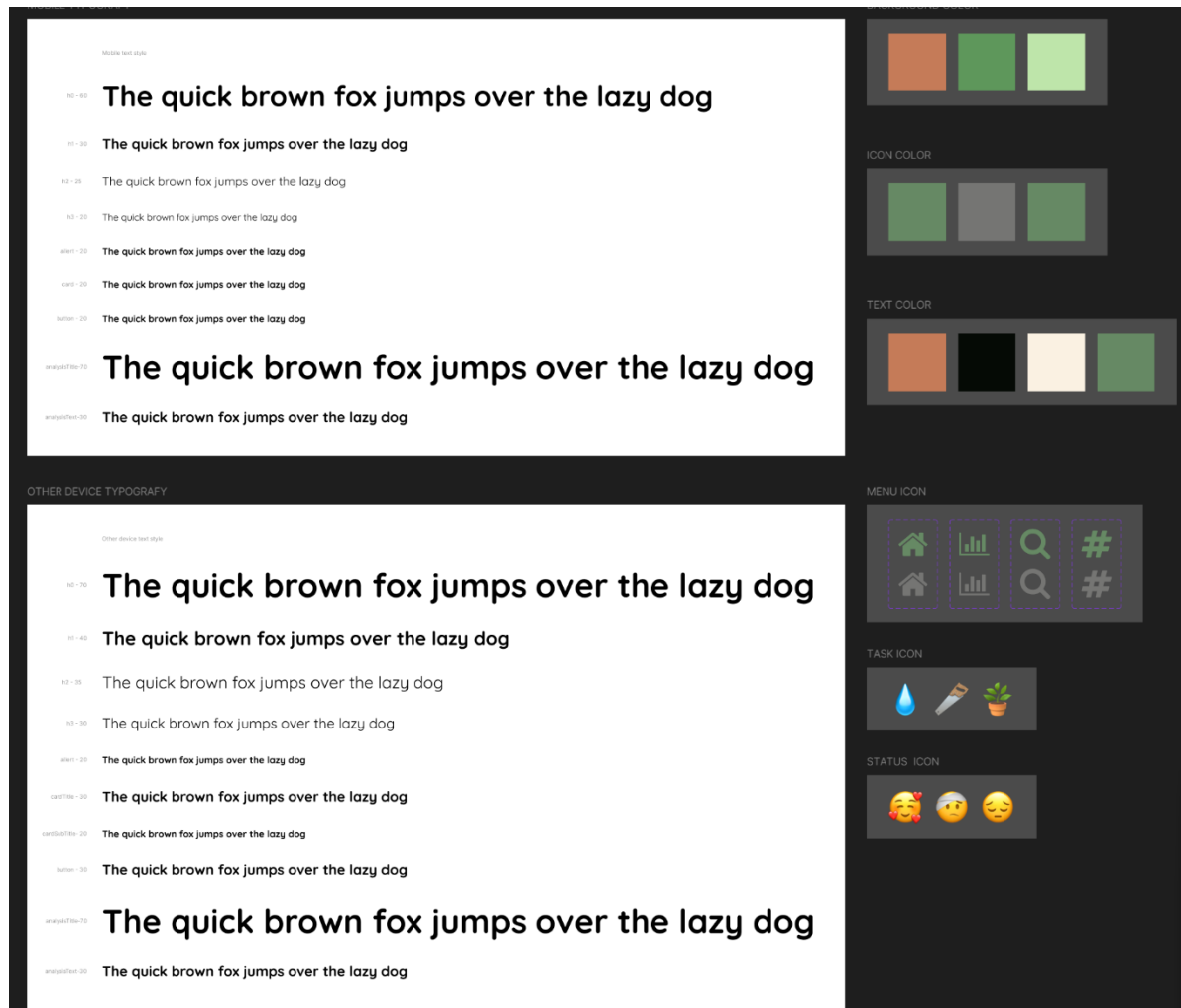
**Tutte le funzionalità specifiche**, come l'aggiunta o la modifica di una pianta, la visualizzazione dei dettagli o la registrazione delle attività di cura, **sono state distribuite** all'interno di queste pagine in modo coerente, così da garantire un'esperienza fluida e intuitiva per l'utente.

## Sviluppo

### UI

Per la progettazione grafica dell'app, è stato utilizzato **Figma**, uno strumento professionale molto diffuso per la realizzazione di interfacce utente e prototipi interattivi. L'utilizzo di Figma ha permesso di affrontare la progettazione in modo strutturato, partendo inizialmente dalla realizzazione di un **wireframe**.

In parallelo alla realizzazione del wireframe, sono stati già definiti alcuni aspetti stilistici come la **tipografia**, la **palette di colori** e altri elementi grafici come le **icone**, al fine di garantire fin da subito coerenza visiva e identità all'interfaccia.



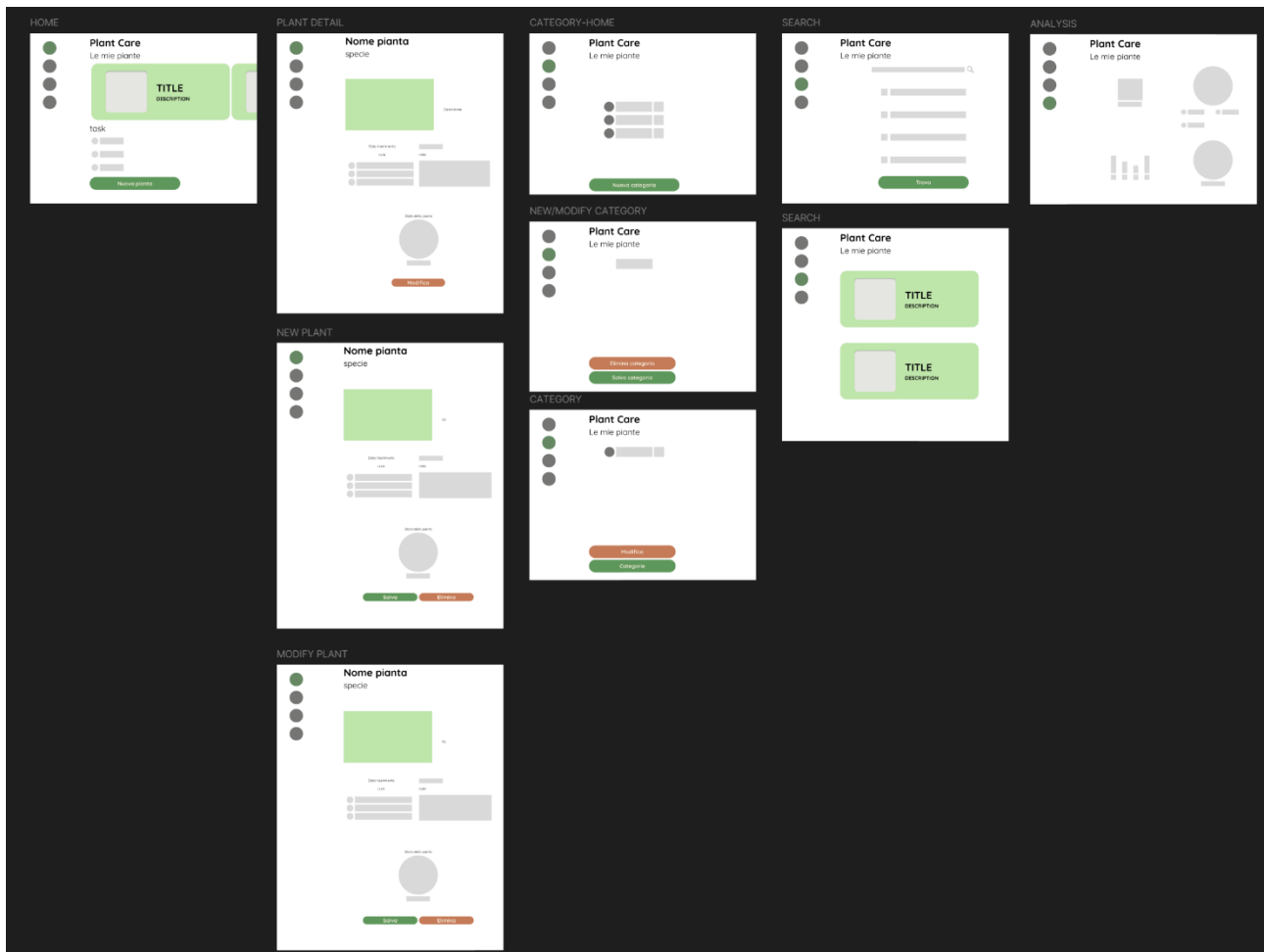
*Tipografia scelta e palette colori e icone*



Wireframe versione mobile

Il wireframe ha rappresentato la prima fase di definizione dell'interfaccia, con l'obiettivo di stabilire la disposizione degli elementi principali e il flusso di navigazione tra le diverse schermate dell'app. In questa fase, l'attenzione si è concentrata soprattutto sull'organizzazione logica delle funzionalità, senza ancora entrare nei dettagli stilistici. Sono state così delineate le quattro pagine principali dell'applicazione: **Home**, **Categorie**, **Ricerca** e **Statistiche**, assegnando a ciascuna un ruolo specifico nella gestione e consultazione delle informazioni.

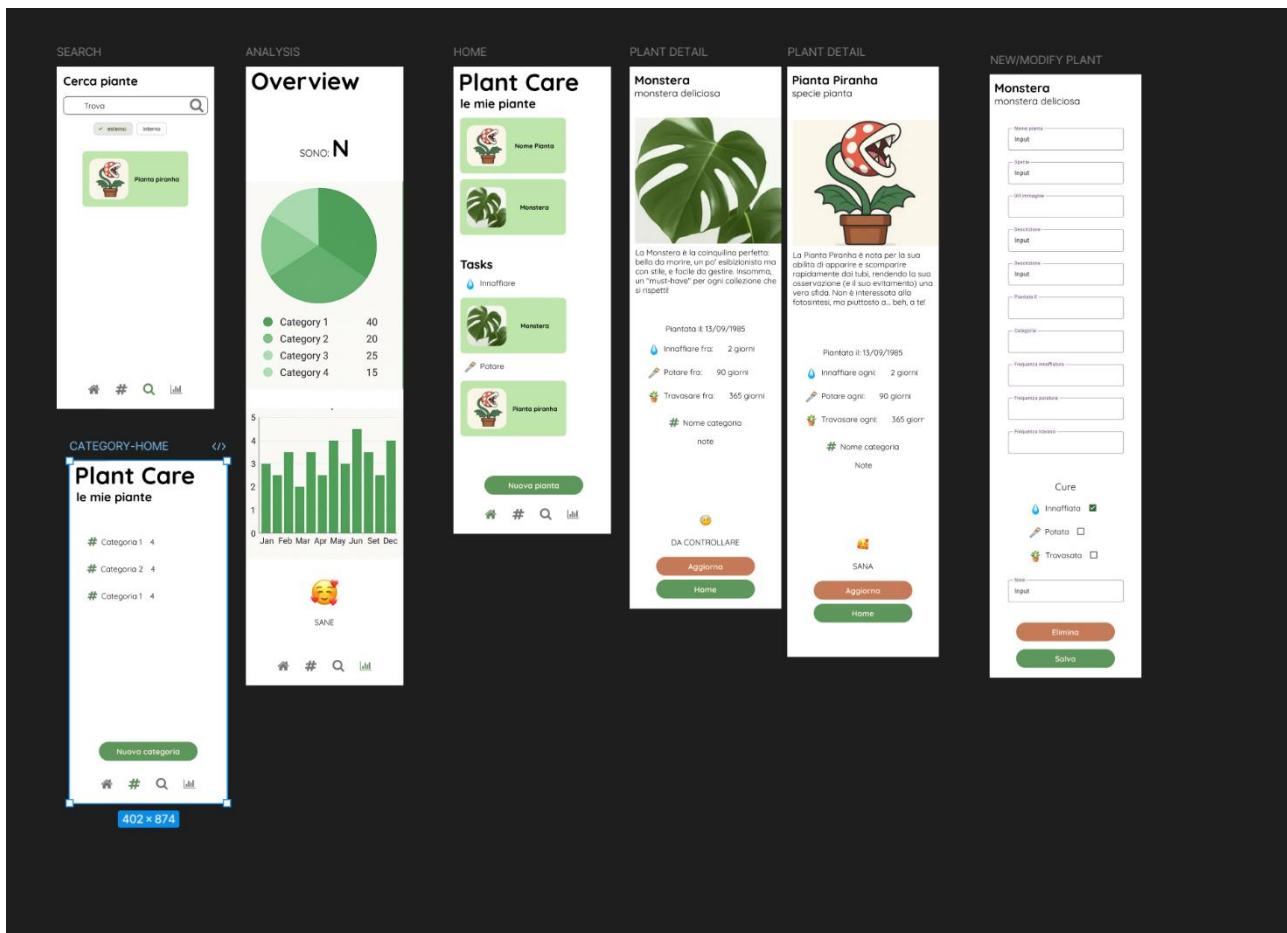
Con la realizzazione del primo wireframe, è stata successivamente delineata anche una versione adattata per i diversi dispositivi, tenendo conto delle variazioni di proporzioni e dimensioni degli schermi.



### *Wireframe versione multidispositivo*

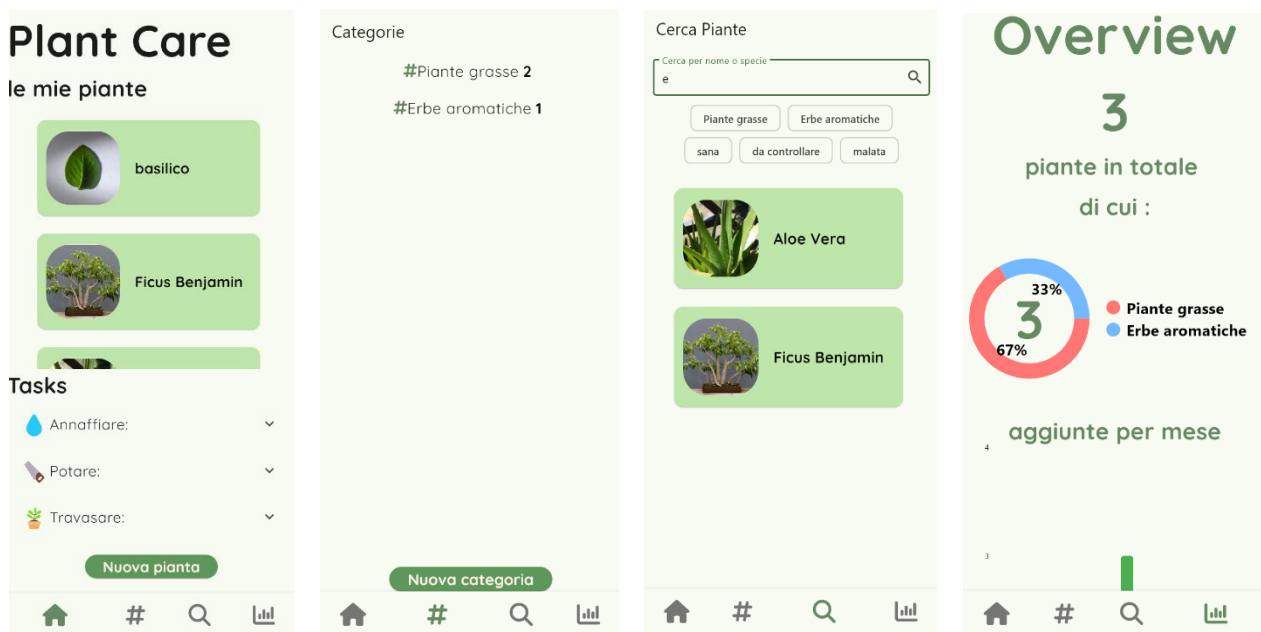
Una volta definita la struttura generale attraverso il wireframe, si è passati alla creazione del **mockup**, ovvero una versione più avanzata e dettagliata delle schermate. In questa fase è stata definita la palette di colori, la tipografia, le icone e lo stile visivo complessivo dell'app, con l'obiettivo di creare un'interfaccia non solo funzionale ma anche esteticamente coerente e gradevole. Il mockup ha permesso di visualizzare in modo realistico l'aspetto finale dell'app e di simulare l'esperienza d'uso, facilitando eventuali revisioni prima di procedere allo sviluppo vero e proprio.





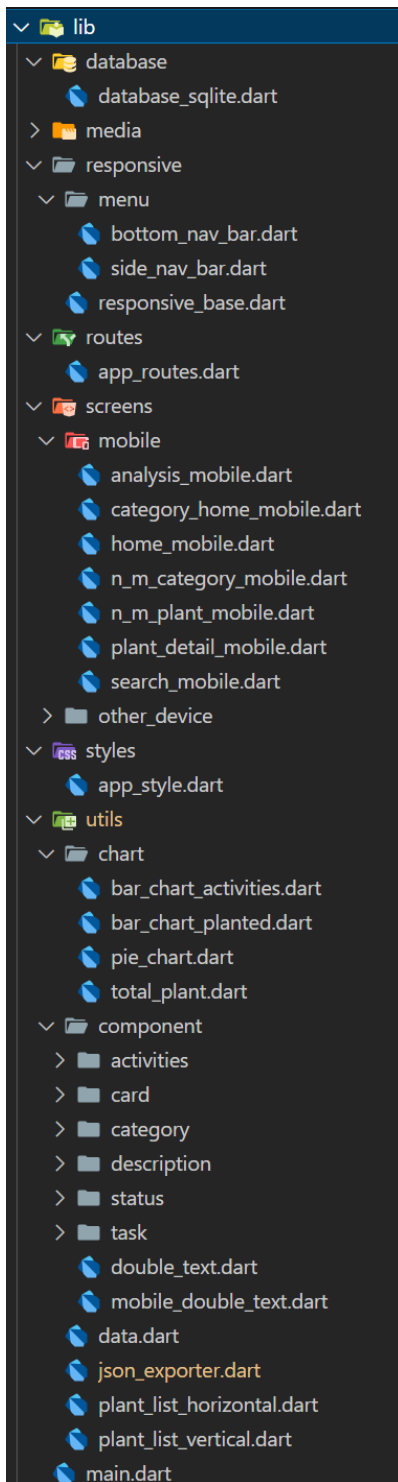
Prima versione dettagliata dell'app e delle sue componenti

L'approccio iterativo consentito da Figma ha reso semplice il passaggio dal concept alla progettazione definitiva, offrendo una visione chiara e condivisibile sia per la parte grafica che per la definizione delle funzionalità, in modo da avere un riferimento solido durante la fase di sviluppo con Flutter.



Versione mobile definitiva delle schermate principali

## Struttura delle cartelle e organizzazione del codice



Tutta l'implementazione dell'app risiede nella cartella principale **lib**, dalla quale, attraverso il file **main.dart**, viene avviata l'app Flutter. Il codice è organizzato in sottodirectory tematiche, ciascuna denominata in modo descrittivo per facilitarne la comprensione e la manutenzione.

In primo luogo, la cartella **responsive** si occupa di rilevare il tipo di dispositivo (mobile, tablet, desktop) e di generare il widget base che adatta automaticamente il layout. Una volta stabilita la piattaforma, le diverse **schermate** dell'app come: home, categorie, ricerca e statistiche vengono caricate dalla directory **screens**, suddivisa a sua volta in sottocartelle (**mobile**, **other\_device**) per gestire le specificità di ciascun form factor.

Tutti i piccoli widget e le utility che compongono l'interfaccia sono raccolti nella cartella **utils/component**. Qui trovano posto i componenti riutilizzabili e le funzioni di supporto utilizzate in tutta l'applicazione.

La cartella **utils** non ospita soltanto componenti riutilizzabili, ma include anche diversi file di supporto fondamentali, tra cui:

- Il **meccanismo di esportazione in JSON (json\_exporter.dart)**, responsabile della generazione automatica del file di backup ad ogni modifica al database.
- Un **file JSON di test (data.dart)**, contenente esempi predefiniti di piante da caricare per verificare il corretto funzionamento dell'app in fase di sviluppo.
- La sottocartella **chart** che raggruppa utility specifiche per la visualizzazione dei grafici.

Questa organizzazione garantisce che tutte le funzionalità ausiliarie e i dati di prova siano centralizzati in un unico modulo di utilità.

Altri moduli chiave comprendono:

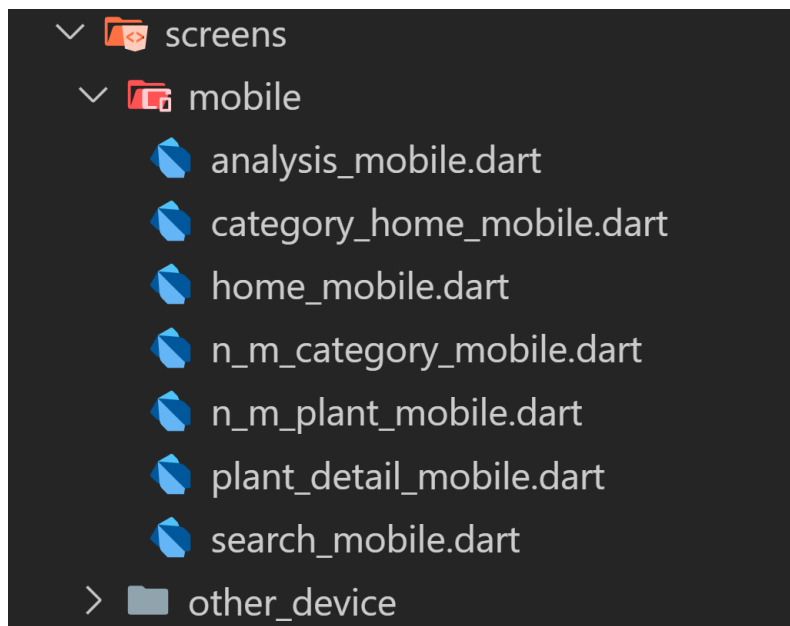
**database**, che centralizza l'accesso a SQLite e le operazioni CRUD;

**styles**, che contiene definizioni di tema, colori e tipografia;

Questa struttura modulare garantisce una chiara separazione delle responsabilità, semplifica il riutilizzo dei componenti e rende l'app facilmente estendibile.

*Struttura ad albero della directory /lib*

## Screens/mobile

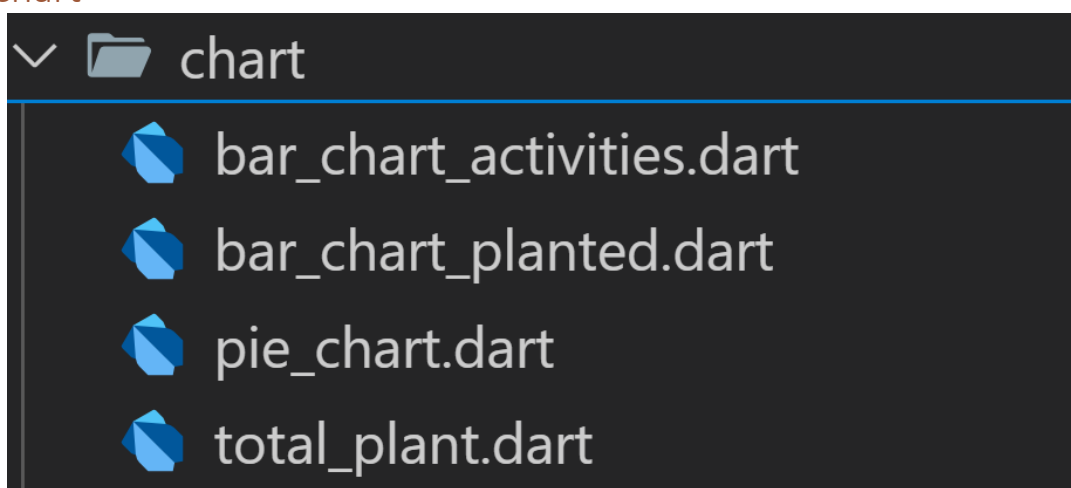


*Struttura ad albero della directory screens/mobile*

Nella directory **screens/mobile** sono raccolte tutte le implementazioni delle schermate ottimizzate per dispositivo mobile. Ogni file Dart corrisponde a una specifica pagina (ad esempio Home, Categoria, Dettaglio pianta, Ricerca, Analisi) e i nomi, volutamente autoesplicativi, facilitano la comprensione del ruolo di ciascun componente. Questa organizzazione rende il codice più leggibile, ne semplifica la manutenzione e ne agevola l'estendibilità futura.

La cartella **other\_device** contiene gli stessi file, opportunamente adattati per la visualizzazione su dispositivi diversi. Questa duplicazione è intenzionale, poiché facilita l'eventuale concentrazione dello sviluppo su una specifica tipologia di dispositivo.

## Utils/chart

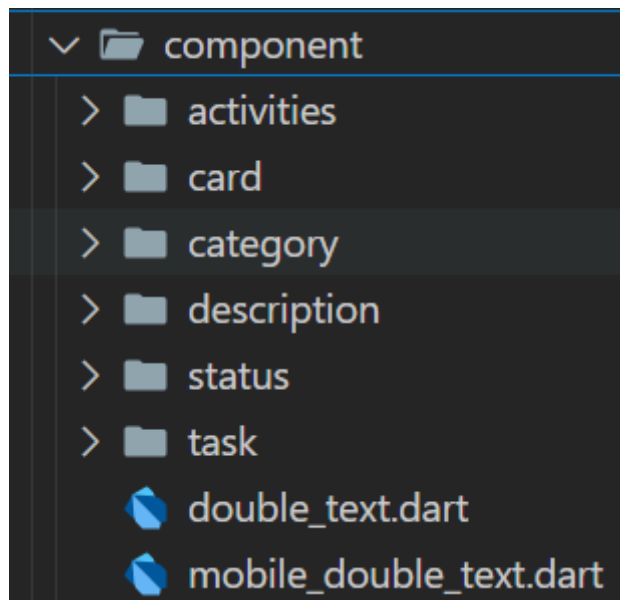


*Struttura ad albero della directory utils/chart*

La cartella raccoglie tutte le utility dedicate alla schermata di analisi. I file presenti in questa cartella includono tutte, o quasi tutte, le dipendenze necessarie per la creazione e gestione dei grafici, grazie all'utilizzo di librerie esterne come **fl\_chart** e **pie\_chart**.

La gestione dello **stato generale della collezione** è stata invece implementata separatamente nella sezione dei **componenti**. Questo perché esistono già dei componenti riutilizzabili per la gestione dello stato, che vengono impiegati anche in altre schermate, e non solo nella sezione di analisi statistica.

## Utils/component



*Struttura ad albero della directory utils/component*

La cartella **utils/component** raccoglie tutti i piccoli widget riutilizzabili che compongono nei dettagli l'applicazione. Ogni sottocartella contiene volutamente due versioni dello stesso codice: una specifica per la parte mobile e un'altra pensata per la gestione multi-dispositivo.

## Descrizione del comportamento

Il comportamento dell'applicazione è uniforme tra la versione mobile e quella per altri dispositivi: la logica sottostante alla gestione della collezione di piante rimane infatti la stessa, mentre a variare è esclusivamente l'aspetto grafico e la disposizione degli elementi, adattata ai diversi formati di schermo.

All'avvio, l'app mostra la **schermata Home**, concepita per guidare immediatamente l'utente all'interazione con l'applicazione. La schermata è organizzata in tre sezioni principali:

- La parte superiore visualizza le **ultime piante aggiunte**, offrendo un rapido colpo d'occhio sulle novità della propria collezione.
- La sezione centrale propone un **promemoria delle attività di cura imminenti**, come innaffiatura, potatura o rinvaso, così che l'utente sia sempre aggiornato sulle necessità delle proprie piante.
- Infine, in basso, è presente una **call to action** ben visibile che invita ad **aggiungere una nuova pianta**. Premendo questo pulsante, si accede a una schermata dedicata dove, tramite un form, è possibile inserire tutte le informazioni necessarie sulla nuova pianta.

Durante l'inserimento, alcuni campi sono stati definiti come obbligatori in fase di progettazione e devono necessariamente essere compilati per poter salvare la nuova pianta.

In qualsiasi momento è possibile annullare l'inserimento e tornare alla Home.

Tornando alla schermata iniziale, le piante registrate vengono visualizzate come **card descrittive**, ciascuna con il nome della pianta e la relativa immagine (se caricata). Toccando una card, l'utente viene indirizzato alla **schermata di dettaglio della pianta selezionata**, dove può consultare tutte le informazioni inserite e quelle generate automaticamente dall'app (come la prossima attività di cura prevista o lo stato della pianta).

Da questa schermata di dettaglio è possibile:

- **Modificare i dati** associati alla pianta.
- **Registrare le attività di cura svolte**, aggiornando così lo stato della pianta e il database interno.
- **Eliminare la pianta**, se necessario.

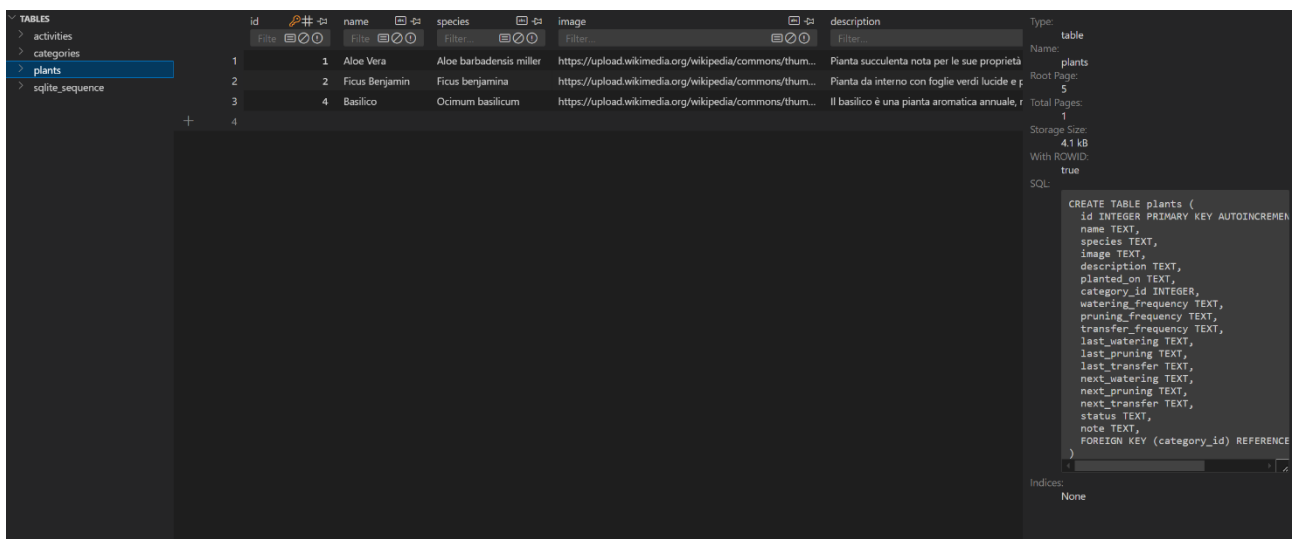
Ogni modifica può essere confermata tramite il salvataggio, oppure annullata tornando indietro tramite la freccia di navigazione posta in alto accanto al nome della pianta.

Oltre alla Home, l'app presenta altre tre schermate principali:

1. **Schermata Categorie:** mostra il numero di piante per ciascuna categoria esistente (ad esempio, piante grasse, da esterno, da interno). Da qui è possibile **creare nuove categorie** o **modificare quelle esistenti**, evitando così di dover aggiornare manualmente ogni singola pianta.
2. **Schermata di Ricerca:** probabilmente la più complessa e completa, permette di **cercare una pianta specifica** per nome o specie, oppure di applicare **filtri** per categoria o per stato della pianta (sana, da controllare, malata). Questo strumento rende rapida ed efficace la navigazione anche nelle collezioni più ampie.
3. **Schermata Statistiche:** offre una **panoramica analitica della collezione**, con grafici che mostrano la distribuzione delle piante per categoria, lo stato generale della collezione e l'andamento delle attività di cura svolte nel tempo.

Questa suddivisione in pagine rende l'esperienza d'uso fluida e organizzata.

## Gestione dei dati



The screenshot displays the SQLite database interface. On the left, a sidebar shows a list of tables: 'activities', 'categories', 'plants' (selected), and 'sqlite\_sequence'. The main area shows the 'plants' table with columns: id, name, species, image, and description. The table contains four rows of data. On the right, a panel shows the table's metadata, including its name, root page, total pages, storage size, and the SQL CREATE statement.

id	name	species	image	description
1	Aloe Vera	Aloe barbadensis miller	https://upload.wikimedia.org/wikipedia/commons/thum...	Pianta succulenta nota per le sue proprietà
2	Ficus Benjamin	Ficus benjamina	https://upload.wikimedia.org/wikipedia/commons/thum...	Pianta da interno con foglie verdi lucide e f...
3	Basilico	Ocimum basilicum	https://upload.wikimedia.org/wikipedia/commons/thum...	Il basilico è una pianta aromatica annuale, r...
4				

Table Metadata:

- Name: plants
- Root Page: 5
- Total Pages: 1
- Storage Size: 4.1 kB
- With ROWID: true

SQL:

```
CREATE TABLE plants (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT,
  species TEXT,
  image TEXT,
  description TEXT,
  planted_on TEXT,
  category_id INTEGER,
  watering_frequency TEXT,
  pruning_frequency TEXT,
  transfer_frequency TEXT,
  last_watering TEXT,
  last_pruning TEXT,
  last_transfer TEXT,
  next_watering TEXT,
  next_pruning TEXT,
  next_transfer TEXT,
  status TEXT,
  note TEXT,
  FOREIGN KEY (category_id) REFERENCES categories
)
```

Indices: None

Database SQLite dell'applicazione

La gestione dei dati dell'applicazione è centralizzata all'interno della classe **PlantCareDatabase**, situata nel file **database\_sqlite.dart**. Questa classe segue il **pattern Singleton**, garantendo che l'applicazione utilizzi un'unica istanza di database durante l'intera esecuzione.

Il database utilizzato è **SQLite**, grazie all'integrazione della libreria **sqflite**, ed è inizializzato con il nome **plants.db**. La struttura del database viene definita nel metodo privato `_createDB`, che crea tre tabelle principali:

- **categories**: contiene le categorie di piante, con un identificativo univoco e il nome (unico).
- **plants**: raccoglie tutte le informazioni su ciascuna pianta, tra cui nome, specie, immagine, descrizione, date di cura (ultimo e prossimo intervento di innaffiatura, potatura, rinvaso), stato della pianta, note e frequenze di cura.
- **activities**: registra le attività di cura svolte su ciascuna pianta, salvando il tipo di attività e la data di esecuzione.

La classe mette a disposizione diversi metodi suddivisi per funzionalità:

- **Inserimento**:
  - `insertPlant`: per inserire una nuova pianta.
  - `insertCategory`: per aggiungere una nuova categoria.
  - `insertActivity`: per registrare una nuova attività di cura.
- **Recupero dati**:
  - Metodi per ottenere una pianta o una categoria specifica (`getPlant`, `getCategory`).
  - Metodi per ottenere tutte le piante, categorie o attività (`getAllPlants`, `getAllCategories`, `getAllActivities`).
- **Modifica**:
  - `updatePlant`: aggiorna i dati di una pianta.
  - `updateCategory`: modifica il nome di una categoria.
- **Eliminazione**:
  - `deletePlant`: elimina una pianta dal database.
  - `deleteCategory`: elimina una categoria.

Sono inoltre presenti query più avanzate che servono per le funzionalità analitiche e per la gestione delle scadenze:

- **Conteggi**:
  - `getTotalPlantCount`: restituisce il numero totale di piante registrate.
  - `getPlantCountByCategoryName`: restituisce il numero di piante appartenenti a una specifica categoria.
  - `getPlantCountByStatus`: restituisce il numero di piante per ciascuno stato (sano, malato, ecc.).
- **Scadenze imminenti**:
  - `getUpcomingWaterings`: elenca le piante che devono essere innaffiate a breve.
  - `getUpcomingPrunings`: elenca le piante che necessitano di potatura imminente.
  - `getUpcomingTransfers`: elenca le piante prossime al rinvaso.

Queste query sono cruciali per alimentare la home page, che notifica le cure più urgenti da svolgere.

- **Statistiche e visualizzazioni grafiche**:
  - `getPlantsPerMonth`: restituisce il numero di piante piantate per ciascun mese, utile per generare grafici temporali.
  - `getCareActivitiesByMonth`: fornisce il numero di attività di innaffiatura, potatura e rinvaso eseguite mensilmente, utile per visualizzazioni comparative.

Infine, la classe offre il metodo **close**, che permette di chiudere la connessione al database quando non è più necessario, evitando sprechi di risorse.

Questa struttura garantisce una gestione solida e performante dei dati, oltre a supportare efficacemente tutte le funzionalità dell'app, sia operative che analitiche.

Non solo: per garantire un ulteriore livello di sicurezza e tracciabilità, **ogni volta che il database viene manipolato**, viene **generato automaticamente un file JSON contenente lo stato aggiornato dei dati**.

Questo file funge da backup immediato e può essere utile sia per finalità di recupero che di analisi esterna.

## Statistiche e Visualizzazioni

Una delle funzionalità chiave sviluppate nell'app è la possibilità di **visualizzare statistiche** in modo chiaro e interattivo, con l'obiettivo di fornire all'utente una panoramica approfondita sia dello stato della propria collezione di piante che delle attività di cura eseguite nel tempo.

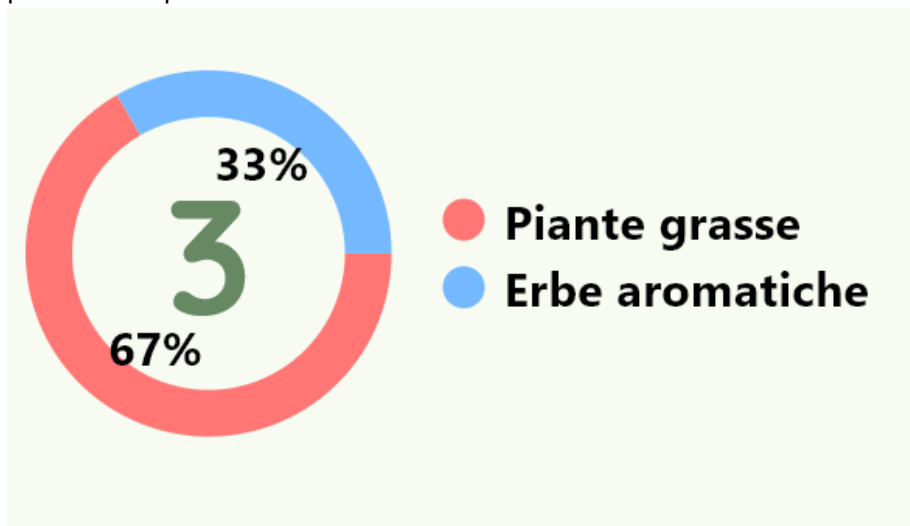
### Tipi di grafici utilizzati

Per rappresentare i dati, sono stati implementati due principali tipi di grafici:

- **Grafico a torta circolare per le categorie di piante:**

Questo grafico mostra la suddivisione delle piante in base alle categorie definite dall'utente. Al **centro del grafico** viene mostrato il **numero totale di piante presenti**, mentre tutto intorno si distribuiscono le diverse porzioni della torta, ognuna colorata in modo distinto per rappresentare una categoria specifica.

Accanto al grafico è presente una **legenda colorata**, che facilita la lettura associando ogni colore alla rispettiva categoria. Questo permette di avere una visione immediata e intuitiva della varietà di piante che si possiedono.

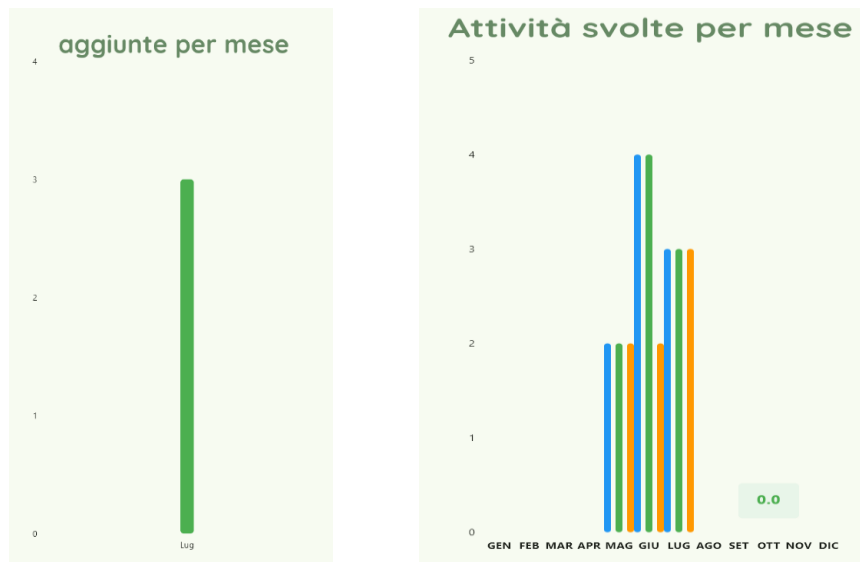


*Grafico rappresentate la distribuzione delle categorie*

- **Grafico a barre per le attività di cura:**

Per monitorare le attività di cura effettuate sulle piante, come innaffiatura, potatura e rinvaso, è stato realizzato un **grafico a barre multiple**. In questo grafico:

- Ogni attività ha un **colore distintivo**.
- L'asse X rappresenta i **mesi dell'anno**, così da permettere all'utente di osservare la frequenza delle attività nel tempo.
- L'asse Y indica la **quantità di interventi** effettuati.  
In questo modo, è possibile verificare facilmente in quale periodo dell'anno si sono concentrate più cure, o quali attività sono state eseguite più frequentemente.



*Grafici a barre riguardanti le attività svolte per relativo mese*

I dati principali che alimentano i grafici sono:

- La **distribuzione delle piante per categoria**, ottenuta interrogando il database locale.
- Il **conteggio mensile delle attività di cura**, aggregate per tipo di attività e mese di esecuzione.

I dati rappresentati derivano da query SQL che elaborano le informazioni memorizzate nel database SQLite e le aggregano nel formato richiesto dai grafici.

## Testing e Debugging

Il processo di testing dell'app è stato condotto principalmente attraverso **test manuali**, simulando l'utilizzo reale dell'applicazione da parte di un utente tipo. Sono state testate tutte le funzionalità principali:

- Inserimento di nuove piante e nuove categorie.
- Modifica di dati già esistenti.
- Registrazione delle attività di cura.
- Visualizzazione dei grafici aggiornati.
- Verifica della correttezza delle statistiche dopo ogni operazione.

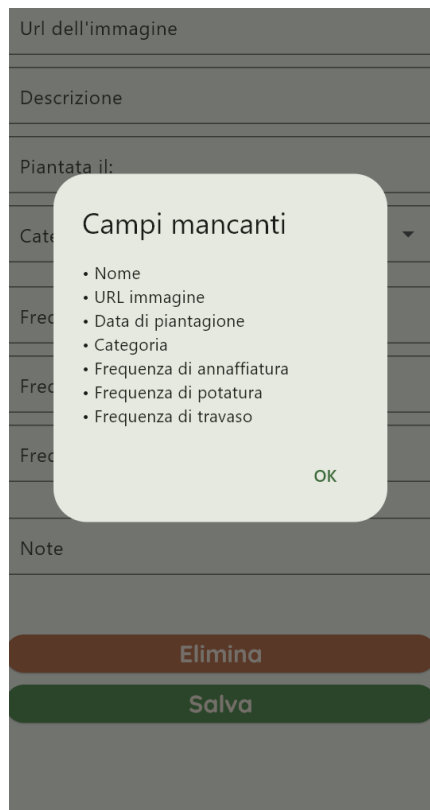
### Errori riscontrati e relative soluzioni

Durante i test sono emersi alcuni **problemi rilevanti**, tra cui:

1. **Errore di caricamento immagini:**  
In fase di inserimento o modifica di una pianta, in alcuni casi l'immagine associata non veniva visualizzata.
2. **Dati errati nelle attività di cura:**  
È stato riscontrato un errore nel **parsing delle date** salvate nel database, il che portava a conteggi non coerenti nei grafici delle attività mensili.
3. **Dati mancanti**



La soluzione è stata quella di guidare l'utente a non commettere questi errori trami l'utilizzo di modali



*Esempio di dati errati o mancanti*

## Conclusioni

La realizzazione dell'app ha rappresentato un percorso formativo e progettuale significativo, che ha permesso di sviluppare un prodotto:

- Funzionale.
- Intuitivo nell'utilizzo.
- Con un valore aggiunto in termini di monitoraggio e organizzazione.

### Valutazione complessiva

L'applicazione soddisfa pienamente gli obiettivi prefissati:

- Gestione ordinata delle piante.
- Tracciamento puntuale delle attività di cura.
- Visualizzazione efficace dei dati attraverso grafici.
- Generazione di **file JSON di backup** dopo ogni modifica al database, per garantire la **sicurezza e la tracciabilità dei dati**.

## Possibili sviluppi futuri

Tra gli sviluppi futuri più rilevanti si prevedono:

- **Introduzione di notifiche push**, per ricordare agli utenti quando una determinata cura è prossima o scaduta.
- **Backup e sincronizzazione in cloud**, in modo da non perdere i dati in caso di cambio dispositivo.
- **Integrazione della fotocamera**, per scattare foto alle piante direttamente dall'app e associarle al loro profilo.
- **Test automatici e unitari**, per garantire un'affidabilità continua del software.

## Difficoltà affrontate e soluzioni adottate

Durante lo sviluppo sono emerse alcune criticità:

- **Gestione delle date in SQLite**: inizialmente complicata per via della necessità di formattazioni coerenti tra inserimento, lettura e visualizzazione. La soluzione è stata standardizzare ogni data su un formato unico e gestire la conversione solo a livello di presentazione.
- **Corretto rendering dei grafici**: la rappresentazione corretta dei dati ha richiesto diversi tentativi di query e trasformazioni, per presentare informazioni affidabili e coerenti nel tempo.
- **Gestione degli errori di percorso immagini**: risolta con fallback predefiniti e validazioni su url.

## Allegati

Tutto il codice sorgente è disponibile su GitHub all'indirizzo:

<https://github.com/Antoniodp087/GRUPPO-22-APPELLO-LUGLIO-2025-MOBPROG>



