Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata (DIEM)

Relazione di progetto

Sviluppo di una applicazione per la gestione di viaggi in React Native

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

Studenti Gruppo 14:

Chirico Emanuel
Matricola: 0612707379

Ciniello Lorenzo
Matricola: 0612708956

Di Carluccio Alessandro
Matricola: 0612707283

Donato Simone
Matricola: 0612707951

Giachetta Corradomaria
Matricola: 0612708054

Docente:
Prof. Francesco Cauteruccio

Last update: 13 luglio 2025

1 Introduzione

MEMOLUGGAGE è un'applicazione mobile multipiattaforma pensata per offrire agli utenti un modo semplice, intuitivo e personalizzabile per documentare, organizzare e analizzare i propri viaggi. Che si tratti di brevi gite fuori porta o lunghi viaggi internazionali, l'app consente di registrare ogni esperienza in modo dettagliato, creando un vero e proprio diario digitale dei propri spostamenti nel tempo.

Vi è la possibilità di creare "entry" di viaggio: ogni voce può contenere il titolo del viaggio, la località visitata, le date di partenza e ritorno, eventuali note personali e una galleria di immagini. Gli utenti possono inoltre indicare se un viaggio è stato particolarmente significativo marcandolo come "preferito" o "da ripetere" per il futuro. Tutti i viaggi aggiunti vengono mostrati in un elenco principale, accompagnato da un accesso rapido per l'aggiunta di nuove esperienze.

Per una migliore organizzazione, è prevista una gestione delle categorie di viaggio: l'utente può associare ogni esperienza a una tipologia (come cultura, natura, relax, ecc.) e creare nuove categorie personalizzate in base alle proprie esigenze. È disponibile anche una sezione di ricerca, che consente di trovare rapidamente viaggi inseriti, filtrandoli per titolo, località, o descrizione.

Uno degli aspetti più interessanti dell'app è rappresentato dalla schermata di analisi, che offre una serie di statistiche utili a comprendere meglio le proprie abitudini di viaggio: numero totale di viaggi effettuati, distribuzione per tipologia, media annuale, e una mappa interattiva che evidenzia le località visitate.

L'app è sviluppata utilizzando tecnologie moderne come React Native, garantendo compatibilità con dispositivi Android e iOS. È progettata per essere responsive, adattandosi fluidamente a diverse dimensioni di schermo e supportando sia l'orientamento verticale che orizzontale. I dati degli utenti vengono salvati in modo persistente, assicurando così che le informazioni restino disponibili anche quando l'app viene chiusa o il dispositivo è spento.

2 Analisi dei Requisiti

Seguono le tabelle contenenti i riquisiti individuati a valle della fase di elicitazione.

2.1 Requisiti Funzionali(RF)

Codice	Nome	Descrizione	Priorità
RF01	Elenco ultimi viaggi	Visualizzazione degli ultimi viaggi aggiunti in ordine cronologico	Must-Have
RF02	Destinazioni preferite	Visualizzazione delle destinazioni contrassegnate come preferite(in rosso)	Must-Have
RF03	Accesso rapido nuovo viaggio	Pulsante o collegamento rapido per aggiungere un nuovo viaggio	Must-Have

Tabella 2.1: Requisiti - Schermata Principale

Codice	Nome	Descrizione	Priorità
RF04	Visualizzazione dettagli	Mostrare titolo, località, date e note personali del viaggio selezionato	Must-Have
RF05	Galleria immagini	Visualizzare le immagini associate al viaggio	Must-Have
RF06	Marcatura viaggio	Permettere all'utente di contrassegnare il viaggio come "preferito" o "da ripetere"	Must-Have

Tabella 2.2: Requisiti - Schermata Dettaglio Viaggio

Codice	Nome	Descrizione	Priorità
RF07	Inserimento titolo	Inserire o modificare il titolo del viaggio	Must-Have
RF08	Inserimento località	Inserire o modificare la località del viaggio	Must-Have
RF09	Inserimento date	Inserire o modificare le date del viaggio	Must-Have
RF10	Inserimento descrizione	Inserire o modificare una descrizione testuale del viaggio	Must-Have
RF11	Inserimento immagini	Caricare o modificare immagini associate al viaggio	Must-Have

Tabella 2.3: Requisiti - Schermata Aggiunta/Modifica Viaggio

Codice	Nome	Descrizione	Priorità
RF12	Elenco tipologie	Visualizzare le tipologie di viaggio (es. cultura, natura, relax, ecc.)	Should-Have
RF13	Conteggio per tipologia	Mostrare il numero di viaggi per ciascuna tipologia	Should-Have
RF14	Creazione categorie	Consentire all'utente di creare nuove categorie personalizzate	Should-Have

Tabella 2.4: Requisiti - Schermata Categorie

Codice	Nome	Descrizione	Priorità
RF15	Barra di ricerca	Consentire la ricerca di viaggi per titolo, località o data	Should-Have
RF16	Filtri avanzati	Possibilità di filtrare i risultati per categoria o preferiti	Should-Have

Tabella 2.5: Requisiti - Schermata Ricerca

Codice	Nome	Descrizione	Priorità
RF17	Statistiche viaggi	Mostrare statistiche su numero di viaggi, categorie più usate, frequenza nel tempo	Nice-to-Have
RF18	Grafici interattivi	Visualizzare grafici (es. a torta o barre) per supportare l'analisi visiva	Nice-to-Have

Tabella 2.6: Requisiti - Schermata Analisi

2.2 Requisiti Non Funzionali(RNF)

Codice	Nome	Descrizione	Priorità
RNF01	Compatibilità dispositivi mobili	L'applicazione deve essere accessibile e pienamente funzionante su smartphone e tablet Android/iOS	Should-Have
RNF02	Persistenza dati	I dati dell'utente (viaggi, immagini, preferiti) devono essere salvati localmente o in cloud in modo persistente	Nice-To-Have
RNF03	Performance	Il caricamento delle schermate principali deve avvenire in meno di 2 secondi	Should-Have
RNF04	Tecnologia di sviluppo	L'applicazione deve essere sviluppata utilizzando React Native o Flutter, per garantire compatibilità multipiattaforma	Must-Have

Tabella 2.7: Requisiti Non Funzionali (Vincoli Tecnici)

3 Struttura e Navigazione dell'Applicazione

Per guidare e ottimizzare il processo di sviluppo dell'app, abbiamo fatto ricorso a dei wireflow, ovvero rappresentazioni visive che uniscono wireframe e flussi di navigazione

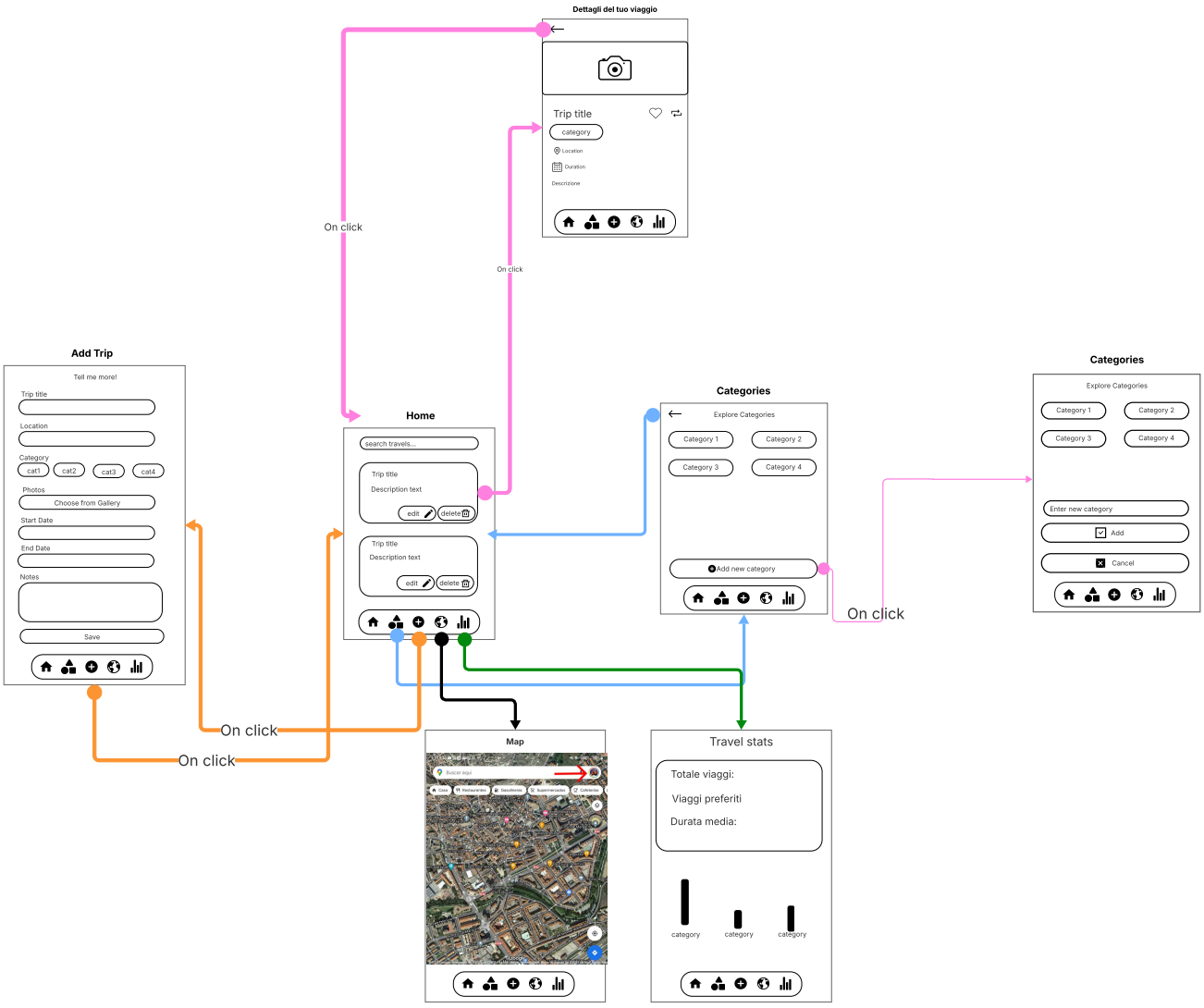

Figura 3.1: Wireflow utilizzato per la progettazione dell'app

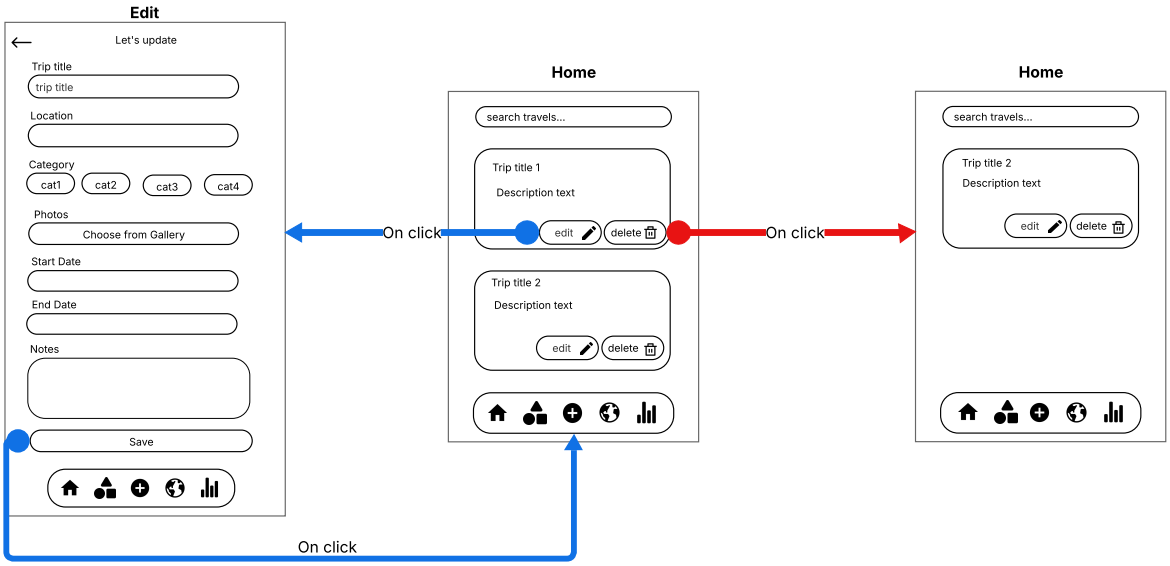

Figura 3.2: Wireflow utilizzato per la progettazione dell'app

Il progetto utilizza **Expo Router** per gestire la navigazione tra schermate, secondo una logica di *file-based routing*. Ogni file all'interno della cartella `app/` rappresenta automaticamente una schermata (o *route*).

3.1 Cartella Principale: `app/`

La cartella `app/` è il punto di ingresso principale per Expo Router. Essa contiene tre elementi fondamentali:

- `_layout.tsx`: definisce il layout principale dell'app, utilizzato per incapsulare tutte le schermate.
- `[edit].tsx`: schermata dinamica per la modifica di un elemento.
- La sottocartella `(tabs)/`: contiene le schermate principali dell'app, accessibili tramite la bottom tab bar.
- La sottocartella `travel`: contiene la schermata per la visualizzazione dettagliata del viaggio `[id].tsx`.

3.2 Navigazione tra Tab: `app/(tabs)/`

All'interno di `(tabs)/` sono presenti tutte le schermate principali, organizzate in modo tale da essere accessibili direttamente dalla bottom tab bar. La struttura attuale è la seguente:

```
app/(tabs)/
 _layout.tsx       // Definisce il componente <Tabs> e la grafica della tab bar
 index.tsx         // Schermata principale (Home)
 add.tsx           // Schermata per aggiungere un elemento
 explore.tsx       // Schermata di esplorazione
 statistics.tsx    // Schermata delle statistiche
 map.tsx           // Schermata con visualizzazione su mappa
```

Il file `_layout.tsx` all'interno della cartella `(tabs)/` configura il componente `<Tabs>` fornito da Expo Router. Questo include:

- Icone e colori delle tab.
- Etichette personalizzate.
- Controllo della visibilità dell'header.

3.3 Schermate Dinamiche e Navigazione Stack

Alcune schermate , come `[edit].tsx` e schermate di errore, sono pensate per essere raggiunte dinamicamente, ad esempio tramite pulsanti, liste o azioni specifiche.

Grazie al supporto di Expo Router, ogni file all'interno della cartella `app/` rappresenta automaticamente una route. La navigazione tra queste schermate è gestita tramite l'hook `useRouter()`, che espone metodi come `router.push(path)` per spostarsi tra schermate, oppure `router.back()` per tornare indietro nella pila di navigazione.

- `router.push(path)`: inserisce una nuova schermata nello stack di navigazione.
- `router.back()`: rimuove la schermata corrente e torna alla precedente.

3.4 Routing Dinamico

Il file `[edit].tsx` utilizza la sintassi tra parentesi quadre per indicare che si tratta di una schermata dinamica. Questo consente di passare parametri direttamente nell'URL, ad esempio:

```
router.push("/edit?id=123")
```

All'interno della schermata, il parametro può essere letto con:

```
const { id } = useLocalSearchParams();
```

Questa tecnica permette di gestire operazioni come modifica, visualizzazione dettagliata o altre interazioni basate su identificatori univoci.

3.5 Persistenza dei Dati

La persistenza dei dati è gestita tramite **PostgreSQL**, un sistema di database relazionale open source, amministrato mediante **pgAdmin**. Il database supporta tutte le funzionalità principali dell'app, tra cui viaggi, tipologie e immagini associate. Di seguito sono riportati gli script di creazione delle tabelle utilizzate.

3.5.1 Tabella `trips`

```
DROP TABLE IF EXISTS public.trips;

CREATE TABLE public.trips (
    id SERIAL PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    description TEXT,
    category VARCHAR(100),
    date TIMESTAMP DEFAULT now(),
    isfavorite BOOLEAN DEFAULT false,
    location TEXT,
    ripeti BOOLEAN DEFAULT false,
    start_date DATE,
    end_date DATE
);

ALTER TABLE public.trips OWNER TO postgres;
```

Listing 1: Creazione della tabella trips

3.5.2 Tabella `tipology`

```
-- Table: public.tipology

-- DROP TABLE IF EXISTS public.tipology;

CREATE TABLE IF NOT EXISTS public.tipology
(
    id integer NOT NULL DEFAULT nextval('tipology_id_seq'::regclass),
    nome character varying(100) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT tipology_pkey PRIMARY KEY (id),
    CONSTRAINT tipology_nome_key UNIQUE (nome)
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.tipology
    OWNER TO postgres;
```

Listing 2: Creazione della tabella tipology

3.5.3 Tabella `trip_images`

```
DROP TABLE IF EXISTS public.trip_images;

CREATE TABLE public.trip_images (
    id SERIAL PRIMARY KEY,
    trip_id INTEGER,
    image BYTEA NOT NULL,
    CONSTRAINT trip_images_trip_id_fkey FOREIGN KEY (trip_id)
        REFERENCES public.trips (id)
        ON UPDATE NO ACTION
        ON DELETE CASCADE
);

ALTER TABLE public.trip_images OWNER TO postgres;
```

Listing 3: Creazione della tabella trip_images

4 Schermate

In questa sezione vengono descritte le schermate principali dell'applicazione. Ogni schermata rappresenta una route accessibile tramite la tab bar o tramite navigazione dinamica.

Le schermate principali dell'applicazione sono:

- **Home** - Schermata principale con l'elenco dei viaggi
- **Aggiungi** - Schermata per aggiungere nuovi viaggi
- **Categorie** - Gestione delle categorie di viaggio
- **Modifica** - Schermata per modificare un viaggio esistente

- **Statistiche** - Visualizzazione delle statistiche sui viaggi
- **Mappa** - Visualizzazione dei viaggi su mappa
- **Visualizza viaggio** - Dettagli di un viaggio specifico

4.1 Home (`index.tsx`)

Nella **Home**, la schermata iniziale dell'applicazione, è possibile visualizzare tutti i viaggi che sono stati aggiunti e salvati, grazie al componente TripCard presente in `/app/components/`; con uno sfondo rosso vengono distinti i viaggi indicati come preferiti. Ogni viaggio inserito può essere eliminato, grazie al pulsante *Delete* in rosso, o modificato, con il pulsante *Edit* in blu, andando quindi nella schermata di **Modifica**. Tramite l'area di testo nella parte superiore si possono filtrare i viaggi per nome e località.

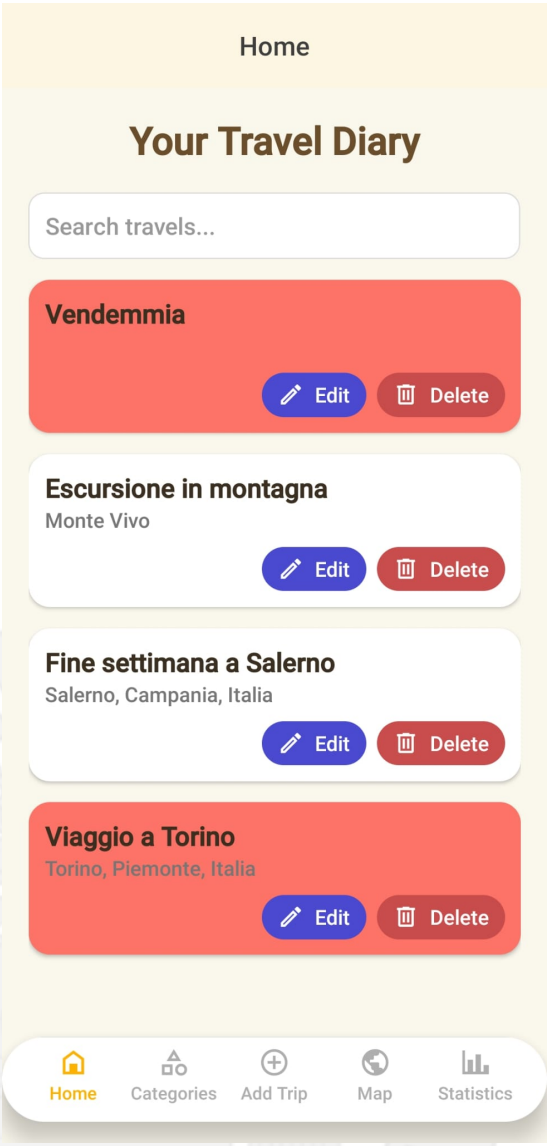

Figura 4.1: Tema chiaro

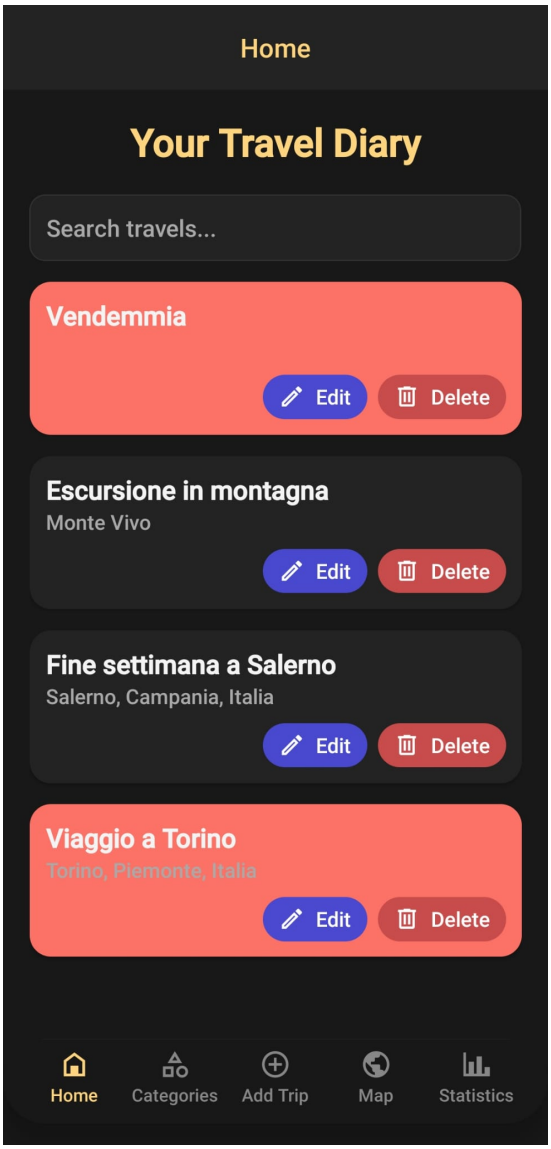

Figura 4.2: Tema scuro

4.2 Aggiungi (`add.tsx`)

In **Aggiungi** si inseriscono i dati dei nuovi viaggi da salvare. I vari campi da poter compilare sono:

- Titolo del viaggio (obbligatorio).
- Località.
- Categoria del viaggio.
- Foto da selezionare.
- Data di inizio viaggio (obbligatoria).
- Data di fine viaggio (obbligatoria).
- Note aggiuntive sul viaggio.

C'è la possibilità di selezionare il luogo con precisione grazie all'API di reverse geocoding di OpenStreetMap, Nominatim, che fornisce dei suggerimenti in base a quello che viene scritto nell'area di testo. Se una categoria viene selezionata cliccandoci sopra, questa può essere deselezionata ricliccando su di essa; inoltre la modifica di una categoria nell'apposita schermata comporta un aggiornamento istantaneo anche in **Aggiungi**. Una volta inserite, le immagini compaiono nella sezione *Photos* e ognuna ha un pulsante che permette di rimuoverla singolarmente dalla selezione in caso di errore. Le date, oltre ad essere obbligatorie hanno un altro vincolo, quella di fine viaggio non può essere antecedente a quella di inizio viaggio; è possibile selezionare date successive a quella corrente per consentire all'utente di inserire anche viaggi non ancora effettuati ma in programma (scrivendolo magari nelle note aggiuntive). Il pulsante *Save* alla fine permette di aggiungere il viaggio tra quelli salvati e svuota i campi della schermata, che rimangono inalterati altrimenti.

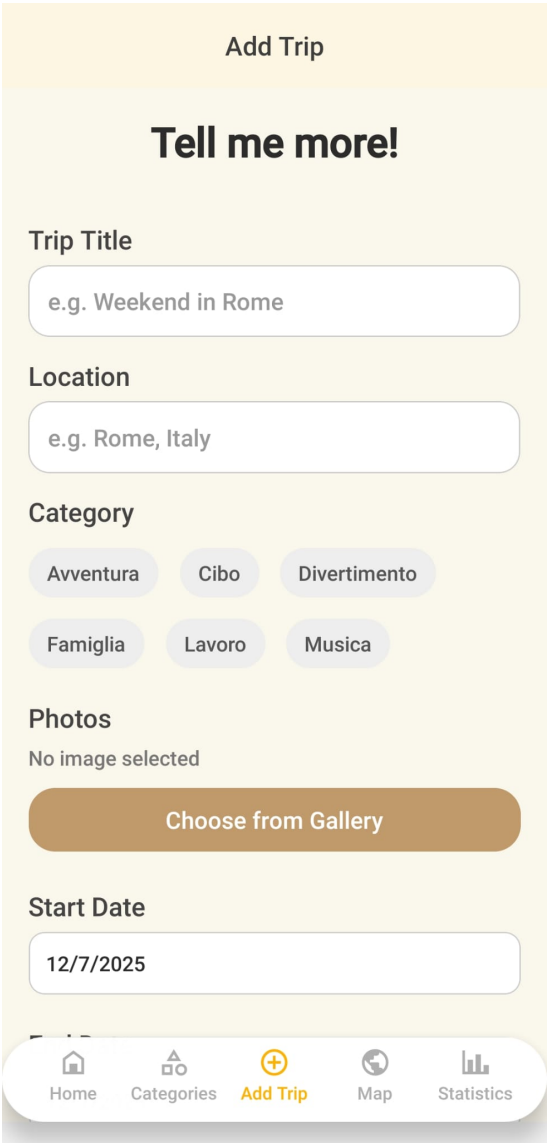

Figura 4.3: Tema chiaro

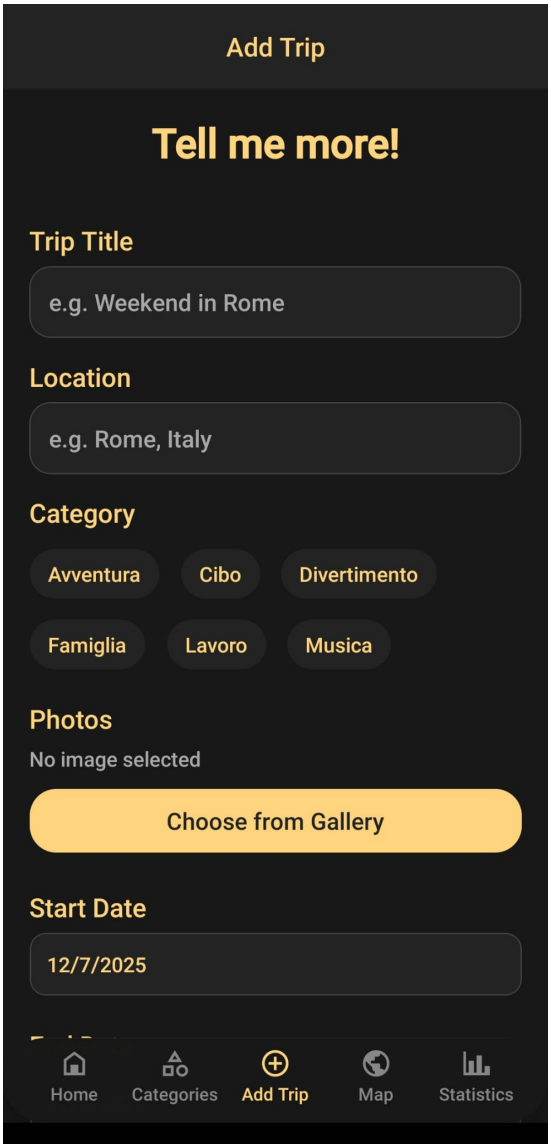

Figura 4.4: Tema scuro

4.3 Categorie (`explore.tsx`)

Nella schermata **Categorie** si possono vedere le varie categorie aggiunte, ognuna con un pulsantino in alto a destra che la elimina. In basso il pulsante *Add new category* fa comparire un'area di testo con un pulsante *Add* per aggiungerne di nuove e *Cancel* per chiudere l'area di testo.

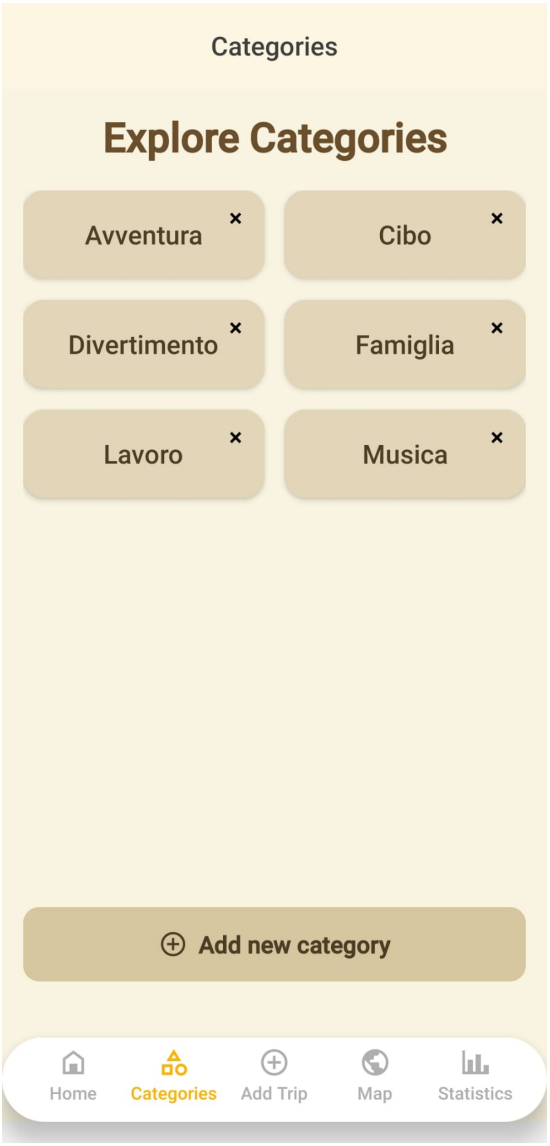

Figura 4.5: Tema chiaro

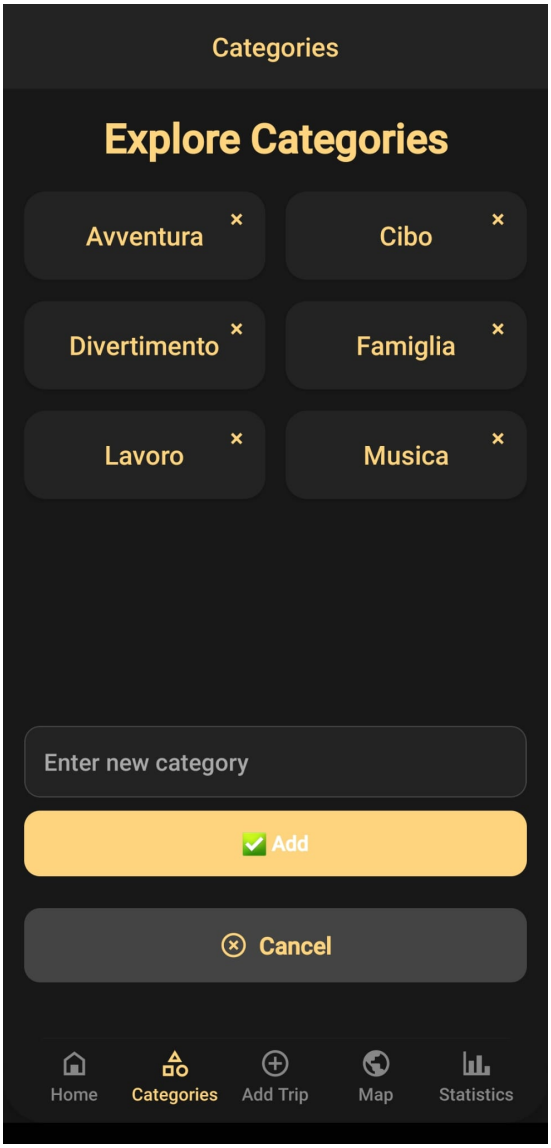

Figura 4.6: Tema scuro

4.4 Modifica (`[edit].tsx`)

Dopo aver premuto il pulsante *Edit* di un viaggio nella schermata **Home**, si arriva alla **Modifica**. I campi vengono compilati automaticamente con le informazioni del viaggio che si vuole modificare e sono gli stessi della schermata **Aggiungi**. La differenza tra questa schermata e le altre sta nel non essere presente nella bottom tab bar, difatti non è possibile muoversi nell'applicazione se non tornando indietro alla **Home** grazie alla freccia in alto a sinistra.

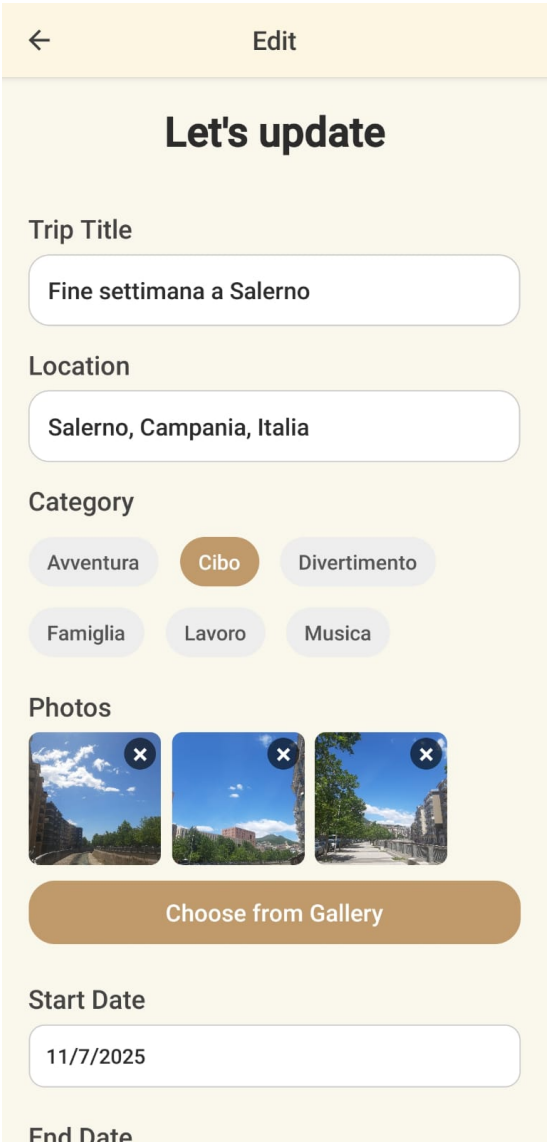

Figura 4.7: Tema chiaro

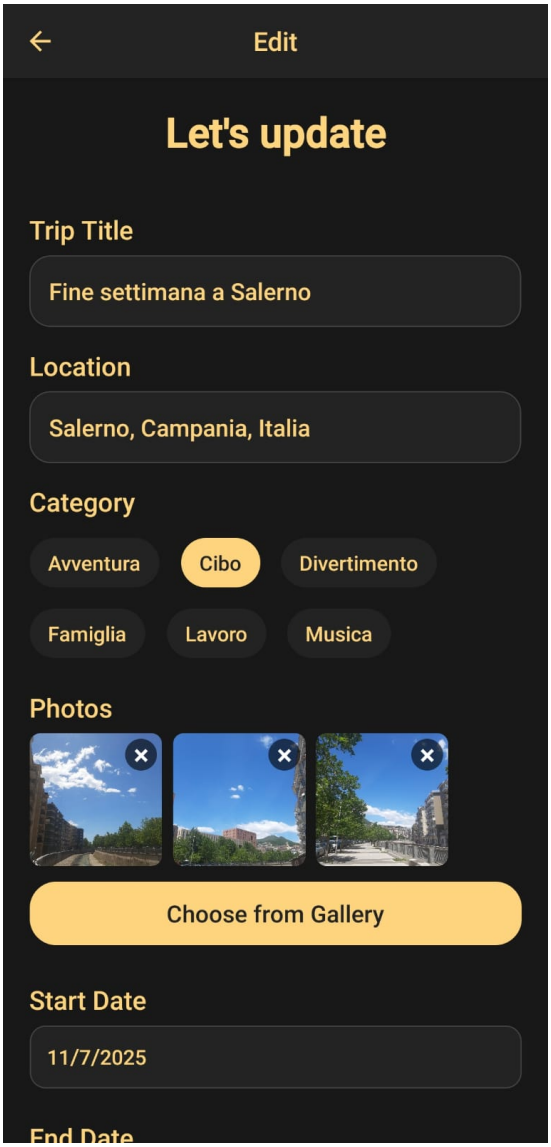

Figura 4.8: Tema scuro

4.5 Statistiche (`statistics.tsx`)

Nella sezione delle **Statistiche** l'utente vede il numero totale dei viaggi che ha aggiunto, quanti ne ha aggiunti tra i preferiti e la durata media dei viaggi salvati. Viene fatta anche una comparazione sulle categorie e sul numero di categorie dei viaggi inseriti.

Figura 4.9: Tema chiaro

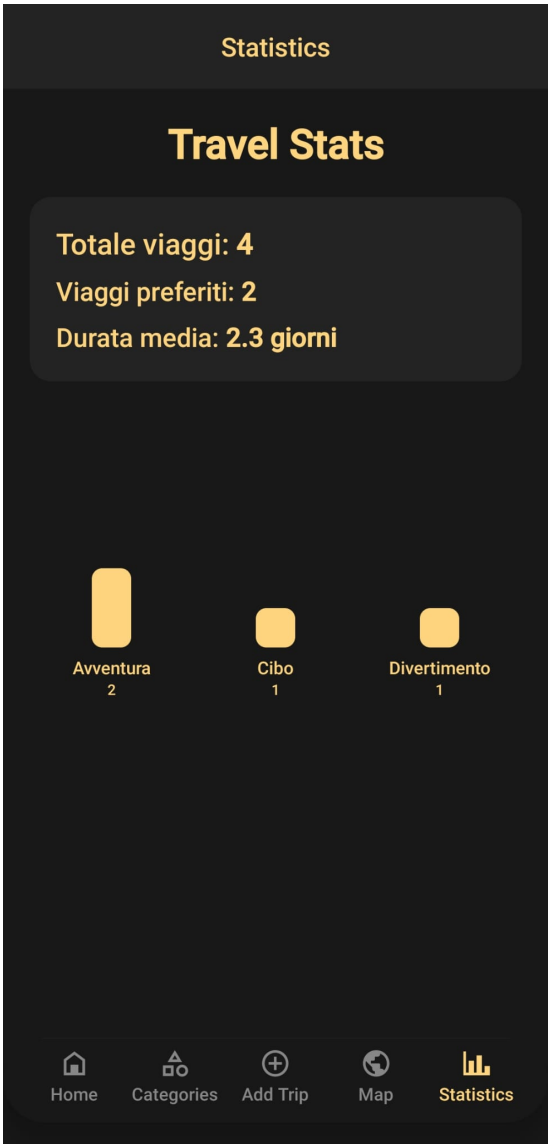

Figura 4.10: Tema scuro

4.6 Mappa (`map.tsx`)

Tra la varie schermate, c'è anche quella della **Mappa**. Sempre grazie a Nominatim di OpenStreetMap, l'applicazione mostra dei marker su una mappa per ogni viaggio aggiunto che ha una località. Schiacciando su un marker vengono visualizzati il nome dato al viaggio e la località inserita. La mappa è centrata inizialmente sull'ultimo viaggio inserito ma è possibile muoversi per controllare ogni angolo del mondo.

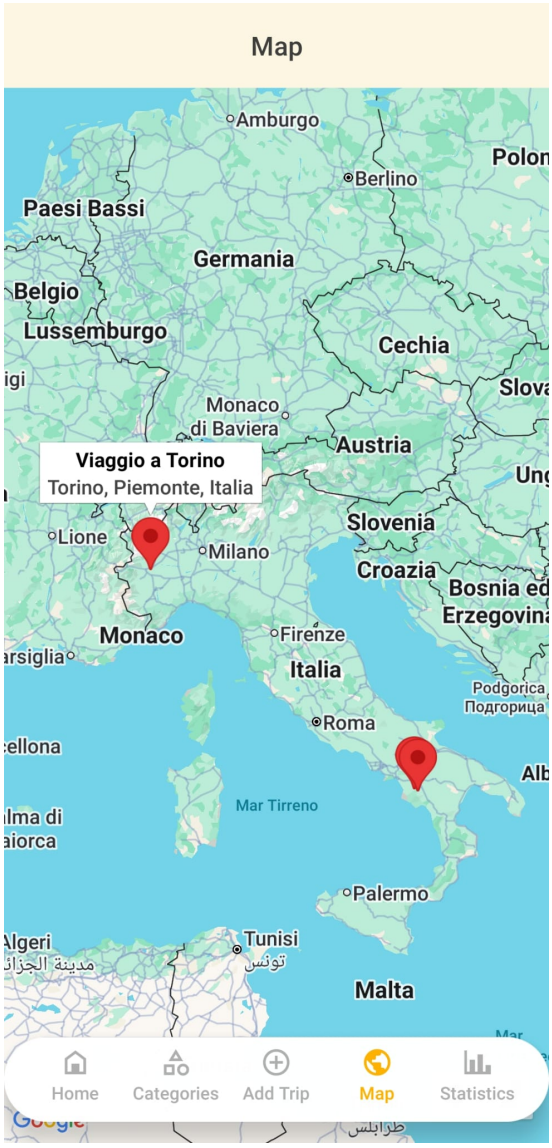

Figura 4.11: * Tema chiaro

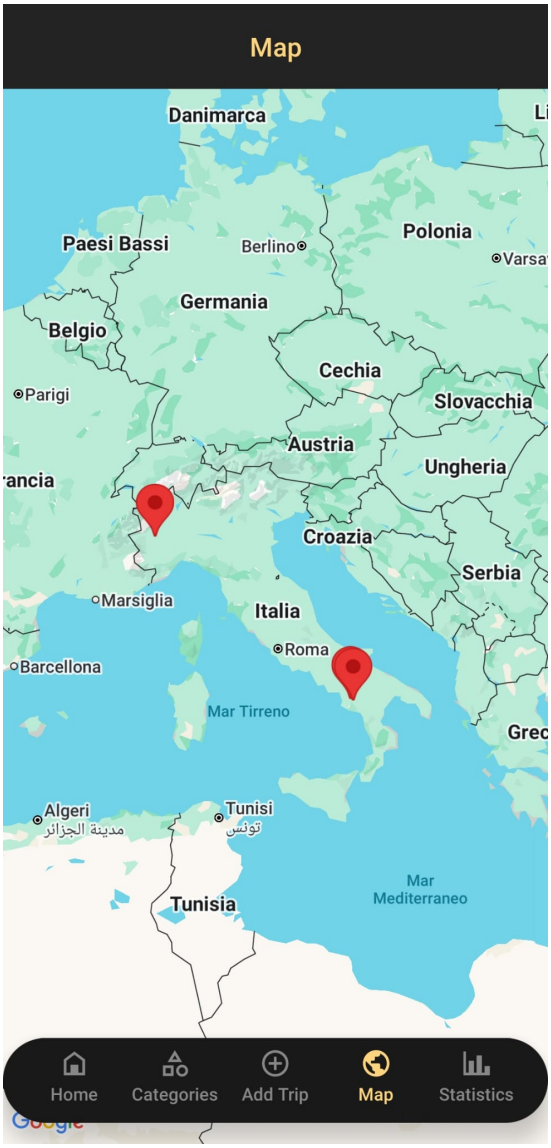

Figura 4.12: Tema scuro

4.7 Visualizza viaggio (`[id].tsx`)

La visualizzazione dettagliata dei viaggi si raggiunge soltanto dalle *TripCard* nella schermata **Home**. Se sono state inserite immagini al momento dell'aggiunta del viaggio, nella zona superiore queste vengono visualizzate; è possibile scorrere a destra e a sinistra nel caso siano più di una e la schermata si adatta dinamicamente in base alle dimensioni delle singole immagini. Subito sotto ci sono le altre informazioni: titolo, categoria, località, date e note aggiuntive. È in questa sezione che si può aggiungere un viaggio tra i preferiti grazie al pulsante a forma di cuore, che si riempie nel caso il viaggio sia già nei preferiti, oppure di indicare il viaggio come da ripetere con il pulsante a forma di "refresh" che si colora di blu. Anche questa schermata come quella di **Modifica** non essendo in `/app/(tabs)/` non ha la bottom tab bar e si può tornare indietro grazie alla freccia in alto a sinistra.

Figura 4.13: Tema chiaro

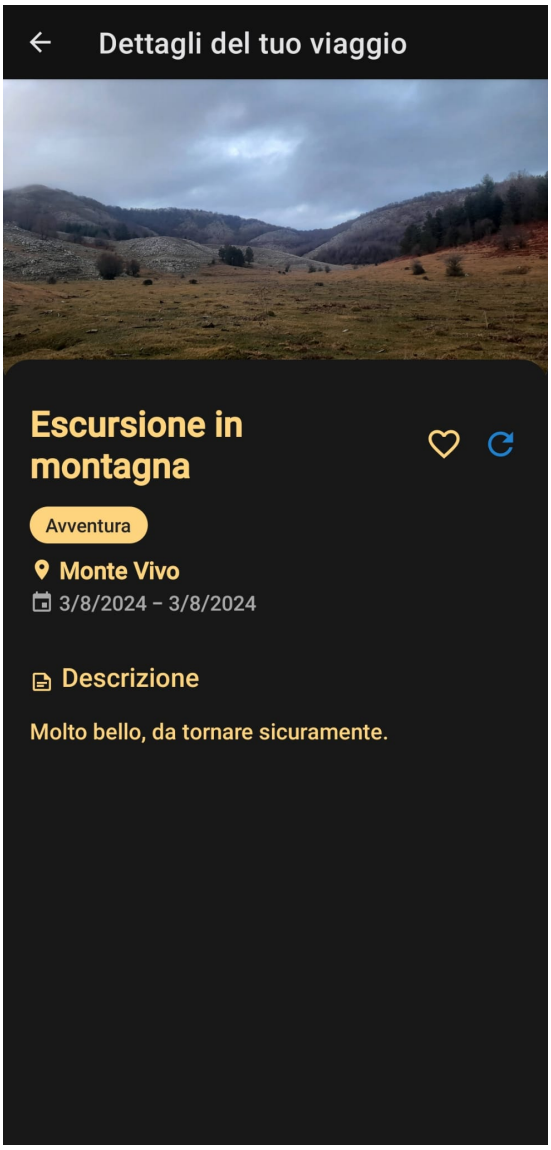

Figura 4.14: *
Tema scuro