Bagnolini Tommaso
Laboratorio di Programmazione di Sistemi Mobili
Mat.: 0001071116

Introduzione

Il progetto che presento consiste nello sviluppo di un'applicazione mobile nativa per Android, chiamata EventRadar. L'app è pensata per aiutare gli utenti a scoprire eventi pubblici, culturali e musicali che si svolgono in Italia, sulla base della località e di una data scelta.

L'obiettivo è stato quello di realizzare un'app che offra all'utente:

- Possibilità di ricercare gli eventi
- Navigazione semplice e intuitiva tra le funzionalità
- Visualizzazione sia testuale che geografica dei risultati
- Gestione personalizzata degli eventi preferiti

Il progetto è stato sviluppato interamente in modalità individuale, prevalentemente per questioni di difficoltà organizzative e di interazione con un ipotetico compagno.

Obiettivi e funzionalità

Partendo dall'idea iniziale indicata nella mail di proposta, ho progressivamente ampliato il progetto, introducendo numerose funzionalità aggiuntive rispetto al piano originale.

Le funzionalità finali implementate sono:

<u>Ricerca eventi:</u>

- Ricerca eventi tramite chiamate API REST.
- I parametri di ricerca sono: città e data.
- La ricerca avviene tramite apposita interfaccia utente con input semplificati:
    - Campo città
    - Campo data con selettore calendario.

<u>Visualizzazione lista eventi:</u>

- Visualizzazione degli eventi in una lista interattiva con card dedicate.
- Ogni card mostra:
    - Immagine evento
    - Nome evento
    - Data
    - Luogo
    - Indicazione se l'evento è già trascorso.
- Gli eventi vengono evidenziati se già avvenuti con apposito badge rosso.
- Filtro "solo eventi futuri" attivabile tramite switch dedicato.

Vista dettagliata evento:

- Schermata di dettaglio che mostra:
  - Immagine
  - Nome evento
  - Data
  - Città
  - Descrizione estesa
  - Link esterno cliccabile
  - Contatti
  - Pulsante per aggiungere o rimuovere dai preferiti.

Salvataggio preferiti:

- Possibilità di aggiungere un evento ai preferiti locali.
- Sezione dedicata ai preferiti, accessibile dalla home.
- Nei preferiti:
  - È possibile ordinare gli eventi per data.
  - È possibile applicare il filtro “solo eventi futuri”.
  - È possibile rimuovere eventi preferiti.
- I preferiti sono salvati in locale tramite Room Database e persistono tra i riavvii dell’app.

Mappa interattiva:

- Visualizzazione degli eventi su una mappa Google.
- Ogni città con eventi ha un marker personalizzato:
  - Se almeno un evento è tra i preferiti, il marker è giallo.
  - Altrimenti, è rosso.
- Ogni marker mostra:
  - Numero di eventi nella città.
  - Dettaglio dei singoli eventi nella città.
- Supporto per la geocodifica dei luoghi tramite Google Geocoding API.
- Auto-zoom sulla città cercata: se l’utente arriva alla mappa dopo aver effettuato una ricerca, la mappa effettua uno zoom automatico sulla città cercata.

Fallback intelligente:

- Se una ricerca non restituisce eventi per una determinata città e data (causa mancanza dati nel database), l’app effettua una chiamata di fallback e propone eventi della stessa città in altre date, tramite un messaggio informativo.

Altre funzionalità

- Gestione dei permessi per la localizzazione.
- Hide automatico della tastiera all’opening dell’app.
- Supporto a eventi multipli nella stessa città.
- Notifica visiva quando la lista è vuota (“Nessun evento trovato”).
- Progress bar e indicatori di caricamento.

Architettura

Il progetto è stato sviluppato seguendo un'architettura MVVM (*Model-View-ViewModel*) per garantire una buona separazione delle responsabilità.

- Model: gestione dei dati e della logica applicativa (Room, Retrofit, Repository).
- ViewModel: logica di interazione tra Model e View.
- View: layout XML + attività e fragment Android.

Tecnologie e librerie utilizzate:

- *Room* per la persistenza locale dei preferiti.
- *Retrofit* per la comunicazione con le API REST.
- *Glide* per il caricamento delle immagini.
- *Google Maps SDK* per la visualizzazione della mappa.
- *Google Geocoding API* per la geocodifica dei luoghi.
- *LiveData* e *ViewModel* per una gestione reattiva dell'interfaccia.

Backend e API

Il backend è stato realizzato in PHP, in ambiente locale, con connessione a un database MySQL su stack XAMPP.

Le principali API implementate sono:

1. `/get_events.php?city=XXX&date=YYYY` → restituisce gli eventi per città e data.
2. `/get_events_by_city.php?city=X` → fallback call: restituisce gli eventi della città senza filtro data.
3. `/get_all_events.php` → restituisce tutti gli eventi.

Il database è composto da una tabella `events` con i seguenti campi:

- id
- name
- date
- location
- imageUrl
- description
- externalLink
- contacts

Il database è inoltre facilmente estendibile per aggiungere nuovi eventi.

Avviare il progetto

Requisiti

- Android Studio (API min 27+)
- XAMPP con Apache e MySQL
- Dispositivo Android o emulatore

Istruzioni

1. Avviare XAMPP e abilitare *Apache* e *MySQL*.
2. Eseguire il file `createDb.sql` fornito con il progetto per creare e popolare il database con la tabella events
3. Copiare la cartella delle API PHP in:

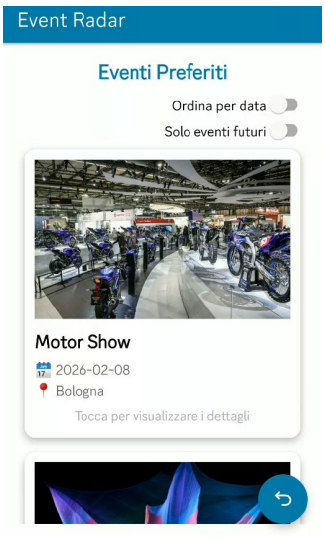
   `…\xampp\htdocs\eventradar-api\`

4. Verificare che le API siano accessibili da browser:

   `http://localhost/eventradar-api/get_all_events.php`

5. Configurare correttamente la `baseUrl` nelle chiamate Retrofit.
6. Avviare l'app su emulatore o dispositivo reale.

Galleria



Conclusioni

Il progetto EventRadar è stato sviluppato seguendo i requisiti previsti dalla traccia d'esame.

Per quanto riguarda i requisiti minimi:

- La struttura del progetto è organizzata per *packages e moduli*, con chiara separazione tra:
  - dati locali/remoti (`data.local`, `data.remote`),
  - repository di interfaccia verso i dati,
  - dominio (`domain.models`),
  - presentazione/UI (`ui` package suddiviso per feature),
  - utilità comuni (`utils`).

- Sono stati impiegati i componenti di lifecycle di Android:
  - `ViewModel` per la logica di business e la gestione dello stato.
  - `LiveData` per l'osservazione dei dati nella UI.

- L'uso delle coroutines è stato applicato per eseguire operazioni di I/O (API remote, geocoding, Room) su thread secondari, mantenendo responsivo il main thread.

- I dati vengono salvati in uno storage locale:
  - È stato usato *Room Database* per il salvataggio degli eventi preferiti, con mapping tra entità ed *Event model*.

In relazione ai requisiti opzionali:

1. Sono state implementate 2 chiamate remote base + 1 terza per il fallback in caso di assenza di dati cercati per ottenere:
   - Lista di eventi per città e data
   - Eventi alternativi se nessun evento è disponibile nella data specificata.

2. È stata implementata la gestione dei permessi di localizzazione:
   - L'applicazione richiede e gestisce correttamente i permessi di *ACCESS_FINE_LOCATION* e *ACCESS_COARSE_LOCATION*.
   - Integra una mappa interattiva con Google Maps API e posizionamento dinamico dei marker sulla mappa.
   - I marker supportano clustering di eventi per città.

In aggiunta sono state sviluppate funzionalità extra rispetto ai requisiti minimi, tra cui:

- Gestione eventi preferiti con filtro e ordinamento.
- Filtraggio eventi futuri.
- Visualizzazione eventi scaduti in lista.
- Navigazione dettagliata con pagina di dettaglio evento (con campi aggiuntivi: descrizione, link, contatti).
- Interfaccia responsive e coerente.

La struttura è modulare e facilmente estendibile per ulteriori funzionalità future, come:

- Supporto multilingua
- Autenticazione utente
- Notifiche push per nuovi eventi.
- ...