Relazione Applicazione EventLink

Lorenzo Zanini 226617 Lorenzo Masè 227312

Indice

1 Introduzione

La presente relazione è relativa all'applicazione sviluppata da Lorenzo Masè e Lorenzo Zanini per il progetto di Programmazione Mobile 2024 tenuto dal Professore Roberto Battiti dell'Università degli Studi di Trento.
La relazione si divide in sette sezioni, nelle quali inizialmente racconteremo l'idea di EventLink, per poi spiegare come intendiamo inserirci in un mercato già molto sviluppato. Successivamente passeremo ad un'analisi più tecnica del software.
Infine, verranno presentate un'analisi del testing, delle valutazioni e delle prospettive future dell'applicazione.
Ringraziamo i lettori per l'attenzione.

2 Contesto e Motivazione

La proposta è stata quella di creare un'applicazione che fosse capace di offrire un servizio che soddisfacesse dei bisogni.
L'idea di EventLink parte dal bisogno delle persone di informarsi sugli eventi disponibili in modo rapido e sicuro.
Oggigiorno, la maggior parte delle persone si informa tramite i social media; il nostro obiettivo è quello di dare agli eventi uno spazio dedicato, evitando la dispersione tipica dei social, che ormai hanno fin troppi utilizzi.
I nostri utenti target comprendono persone di tutte le età.
Questo approccio porterebbe a una maggiore partecipazione degli utenti agli eventi organizzati, aumentando i guadagni per le aziende e facilitando la scelta degli eventi stessi per gli utenti.
Possiamo ipotizzare diverse situazioni in cui l'app sarebbe particolarmente utile: ad esempio, gli utenti in vacanza potranno conoscere le possibilità offerte dal luogo in cui si trovano senza dover cercare ogni singolo evento.
Anche uno studente che cambia città trarrebbe vantaggio da questo servizio.
Principalmente, gli utenti avranno modo di essere sempre aggiornati, anche tramite l'utilizzo dei preferiti.

3 Analisi della Concorrenza

Come già accennato, il settore delle app per eventi è già molto saturo, quindi era essenziale per noi differenziarci con uno stile unico e migliorare l'esperienza rispetto alle app più utilizzate.
Abbiamo esaminato diverse app, tra cui Dice e EventBrite; pur essendo molto diffuse, spesso gli utenti si trovano a dover affrontare bug o errori di progettazione.
Inoltre, la schermata principale di molte di queste app può trarre in inganno l'utente, facendogli credere che un evento sia facilmente raggiungibile, per poi deluderlo una volta consultato il navigatore.
Abbiamo notato che alcune informazioni, fondamentali per prendere decisioni, risultano essere secondarie rispetto alla tipologia dell'evento.
Ad esempio, la posizione geografica è un fattore cruciale, soprattutto per chi non ha mezzi di trasporto autonomi.
In queste app, la mappa è spesso relegata a un ruolo secondario rispetto all'elenco degli eventi, il quale talvolta è mal filtrato.
Per risolvere questo problema, ci siamo ispirati all'applicazione PrezziBenzina, la quale mostra immediatamente una mappa geografica con tutti i punti di interesse segnati.
L'unico progetto simile all'idea di EventLink è un sito chiamato EventMap; tuttavia, non offre gran parte dei servizi che abbiamo sviluppato.
Inoltre, trattandosi di un sito web, diventa particolarmente scomodo da utilizzare per un utente mobile che desidera scegliere a quale evento partecipare.
Il principale problema che affronteremo più avanti nella relazione è l'effetto "Networking": un nuovo utente si tratterrà nell'applicazione solo se troverà numerosi eventi inseriti e altri utenti interessati.
Ciò richiede una base di eventi costante, aspetto su cui EventLink deve ancora consolidarsi rispetto alle app concorrenti.

4 Tecnologie Utilizzate

Di seguito è riportata una lista delle tecnologie utilizzate e i loro scopi nello sviluppo di EventLink:

- Il software è sviluppato principalmente in Kotlin, XML e Java.
- L'IDE utilizzato è Android Studio.
- Per il database è stato utilizzato Firebase Firestore Database, un database non relazionale fornito da Google.
- Per il database locale è stato utilizzato Java.
- Per la mappa sono state utilizzate le API di Google Maps.
- Per la gestione delle email è stato utilizzato un server Functions di Firebase scritto in Node.js.
- Per l'inserimento automatico di dati è stato utilizzato uno script in Python.
- Per la gestione del progetto è stato utilizzato GitHub.
- Per la gestione delle immagini è stato utilizzato lo Storage di Firebase.
- Come email ufficiale dell'app è stato utilizzato un account Gmail.
- Per la grafica dell'applicazione sono stati utilizzati XML e file .png.

5 Implementazione del Codice: Aspetti Fondamentali

Il codice dell'app è suddiviso principalmente in tre parti:

1. La cartella **other** contiene file utili per l'applicazione, come classi e funzioni.
2. La cartella **pages** insieme a **MainActivity** contiene i file .kt delle pagine dell'applicazione.
3. La cartella **res** contiene le risorse grafiche e le stringhe utilizzate nell'applicazione.

Inoltre, analizzeremo come è stato utilizzato Firebase Firestore Database e il server Functions.

5.1 other

Di seguito elenchiamo i file più importanti nella prima cartella:

- Il file **Cluster** contiene due classi essenziali per il funzionamento dell'applicazione. La prima è MyClusterItem, una classe che estende la classe ClusterItem fornita da Google Maps. Questa classe è fondamentale per la gestione dei marker sulla mappa dell'applicazione. La seconda classe è CustomClusterRenderer, che estende la classe DefaultClusterRenderer. Questa classe ci consente di personalizzare le funzionalità dei cluster, come ad esempio la personalizzazione delle icone associate ai marker e la modifica del numero minimo di marker necessari per creare un cluster.

- I file **EventoLocale** e **DAOEventoLocale** ci permettono di strutturare un'entità nel nostro database locale. Questa entità, chiamata EventLocale, è utile per salvare gli eventi preferiti in locale, evitando così l'obbligo di registrazione per utilizzare questa funzionalità. Questi file sono correlati a **DatabaseLocale**, il quale crea il nostro database e, tramite la sua funzione getInstance, mantiene i dati salvati in locale.

- Il file **Funzioni** contiene un insieme di funzioni globali utilizzate in diverse classi. Un esempio è la funzione rawJSON, che invia un JSON contenente le informazioni sull'email da inviare, tramite metodo POST HTTPS al nostro server Node.js.

- L'ultimo file è **Evento**; questa classe viene utilizzata per creare una lista degli eventi all'interno del nostro codice. Questa lista è utile per evitare di dover accedere al database ogni volta che abbiamo bisogno di informazioni sugli eventi. Viene generata immediatamente all'avvio dell'applicazione.

5.2 pages + ActivityMain

Nella seconda cartella, **pages** insieme a **ActivityMain**, si trova la parte più estesa del codice. Qui sono presenti i file .kt relativi alle diverse pagine dell'applicazione.

- **MainActivity** è la pagina principale che appare quando viene avviata l'applicazione. Inizialmente viene eseguito un caricamento in cui vengono richiesti i permessi per utilizzare la posizione dell'utente.
  Successivamente viene mostrata la mappa contenente i marker. Questi marker sono caricati tramite la funzione *loadMap*, che ottiene gli eventi dal database e li aggiunge alla mappa. In questa pagina vengono gestite anche le altre 3 pagine principali: Menu, Preferiti e Lista. Nel menu possiamo trovare informazioni sull'applicazione, come FAQ e contatti. Nei preferiti possiamo trovare tutti gli eventi che abbiamo aggiunto ai preferiti tramite la loro icona, mentre nella lista troveremo tutti gli eventi presenti nella piattaforma. Gli eventi nella lista e nei preferiti sono ordinati in base alla distanza dell'utente. Nel caso la posizione non venga fornita, verrà utilizzata una posizione nel centro dell'Italia. Nella sezione mappa, possiamo utilizzare il bottone "filtri" che ci permette di applicare due tipi di filtri: per data (come oggi, domani, questo weekend, ecc.) o per tipo di evento. Quando clicchiamo su un evento, viene mostrata un'anteprima di ciò che rappresenta quell'evento. Da qui è possibile passare alla sua pagina dedicata o spostarsi sulla nostra app di navigazione.

- **Evento**: Questa pagina può essere aperta in 4 occasioni, o tramite le liste presenti in Lista, Preferiti e Prenotazioni, oppure quando andiamo a visualizzare in dettaglio gli eventi sui marker. Questa pagina contiene l'immagine e tutte le informazioni riguardanti l'evento. È possibile, tramite l'icona, aggiungere questo evento ai preferiti e, nel caso l'evento richieda una prenotazione obbligatoria, prenotarlo. Per prenotarlo, l'utente dovrà effettuare l'accesso nell'applicazione per ottenere i suoi dati. Queste prenotazioni saranno possibili solo se l'evento avrà ancora posti disponibili; questo dato viene segnato nel campo *Max_prenotazioni*. Nel caso la prenotazione abbia successo, viene aggiornata la tabella delle prenotazioni con la prenotazione dell'utente e verrà inviata un'email di conferma all'utente.

- **PaginaLogin**, **PaginaSignIn** e **PaginaDimenticata**: Queste 3 pagine vengono utilizzate per gestire le funzionalità di login, registrazione e, nel caso l'utente non ricordi la password, per cambiarla. Nella pagina di registrazione vengono recuperati i dati dell'utente e, dopo aver

verificato che siano stati inseriti correttamente, viene inviata all'utente un'email contenente una password generata casualmente. Questo viene fatto perché consideriamo più sicuro utilizzare password generate casualmente per accedere all'account e perché così ci assicuriamo che l'utente abbia accesso all'email che ha utilizzato. I dati verranno salvati nel nostro database nella tabella *Utenti*, e per la password utilizziamo un sistema di hashing SHA-256.
Successivamente, l'utente può accedere utilizzando la pagina di login. Nel caso in cui un utente dimentichi la password, può utilizzare la pagina Dimenticata, dove inserendo la sua email verrà generata una nuova password e inviata tramite email.

- **PaginaProfilo**: In questa pagina sono presenti due sezioni. La prima, **Prenotazioni**, è una lista contenente tutte le prenotazioni associate al nostro account. Da qui è possibile visualizzare i dettagli dell'evento oppure annullare la prenotazione.
  Nelle **Impostazioni** è possibile trovare la possibilità di cambiare la password, dopo aver inserito la password precedente, la possibilità di effettuare il logout e, infine, di eliminare l'account. Quando l'utente elimina l'account, verranno eliminate anche tutte le prenotazioni a suo nome.

5.3 res

Infine, nella terza cartella, possiamo trovare i file XML delle pagine, le animazioni, le immagini utilizzate per la grafica dell'applicazione e le stringhe salvate.

5.4 Firestore Database

Analizziamo ora il Firestore Database, il quale è gestito utilizzando tre table principali:

- Eventi
- Utenti
- Prenotazioni

5.4.1 Eventi

In questa table possiamo trovare le istanze degli eventi inseriti dalle aziende. Gli attributi sono:

- ID_Evento
- Data
- Descrizione
- ID_Azienda
- Immagine
- Indirizzo
- Max_Prenotazioni
- Ora
- Posizione
- Prenotazione
- Prezzo
- Tipo
- Titolo

La maggior parte di questi attributi sono di tipo stringa, essendo descrizioni dell'evento. Questa Tabella ci permette di gestire i vari eventi.

5.4.2 Utenti

In questa table troviamo la definizione di un utente registrato. Gli attributi sono:

- Email
- Nome
- Cognome
- Data di nascita
- Password
- Telefono

Questa table viene principalmente utilizzata nella gestione del SignIn, LogIn, eliminazione dell'account e cambio password.

5.4.3 Prenotazioni

In questa collezione viene salvata la relazione Utente - Evento, quando l'utente decide di prenotarsi. Gli attributi sono:

- ID_Prenotazione
- ID_Utente
- ID_Evento

5.5 Server Storage

Il server Storage di Firebase è stato utilizzato per gestire le immagini degli eventi. Le immagini vengono caricate in questo server e tramite la classe Glide è possibile, tramite l'url dell'immagine, scaricarla e applicarla ad un ImageView.

5.6 Server Functions

Il server Functions di Firebase è stato utilizzato per gestire delle richieste da parte del client. Quando l'utente effettua una registrazione, si invia un metodo POST al server, il quale invia un'email all'utente che ha fatto richiesta. Lo stesso metodo è stato utilizzato per il cambio password e per la prenotazione di un evento. In tutti i casi vengono effettuati dei controlli sulle email e, in caso non fosse possibile inviarle, il server ritorna un errore che viene gestito dal client.

6 Test e Valutazione dei Risultati

Il testing è stato effettuato su diversi dispositivi, sia sfruttando la virtualizzazione di Android Studio, sia collegando Android Studio ad apparecchi fisici. L'applicazione ha dato riscontri positivi in tutti i casi.
Durante la virtualizzazione, è possibile ottenere degli errori a causa della gestione della memoria durante l'operazione di loadMap. Tuttavia, su dispositivi fisici non sono stati riscontrati problemi.
Le Email vengono inviate ai destinatari correttamente , i controlli di sicurezza restituiscono i valori aspettati e i Database Locale e Firebase, gestiscono adeguatamente ogni operazione di aggiornamento.

7 Conclusioni e Prospettive Future

In conclusione, l'applicazione è funzionante, la logica di ogni operazione è ben gestita e la sicurezza è garantita.
Per il futuro, l'idea è quella di creare una pagina web dove le aziende, previa approvazione da parte degli amministratori, potranno inserire i propri eventi. Il sito web avrebbe unicamente questa funzionalità e non sostituirebbe l'applicazione.
Come menzionato all'inizio, questo tipo di applicazioni soffre dell'effetto "Networking", quindi l'obiettivo sarà quello di contattare aziende interessate a inserire i propri eventi nell'applicazione, per garantire una base stabile di eventi.
In questo modo, una volta inseriti eventi che ricorrono periodicamente, l'applicazione avrebbe possibilità di prendere piede nel mercato. EventLink si propone per un utilizzo congruo con la sua idea iniziale.
Si ringraziano i lettori per l'attenzione.