**RELAZIONE APPLICAZIONE FOUND EVENT**

FoundEvent è un’applicazione di ricerca eventi che permette di ricercare eventi nel mondo, con un sistema di prenotazione biglietti per essi. La ricerca può essere fatta in 2 modi : scrivendo il nome di una località o disegnando su google maps la zona su cui si intende cercare. I biglietti si possono prenotare nella schermata di informazioni dell’evento.

**ARCHITETTURA**

L’utente effettua le operazioni di login/registrazione/prenotazione biglietti tramite una connessione TCP/IP; l’aggiornamento del database interno degli eventi è svolto in modo automatico dopo il login. Il sito di storage usato è jsonbin.io, il quale permette di operare su file json (nominati tramite un ID), sia pubblici (che tutti possono vedere) che privati (visibili solo all’utente, necessitano di una secret key per essere acceduti). Il sito consente anche di raggruppare dei file in delle collezioni (chiamate collections, anch’esse identificate da un ID); i file degli utenti registrati stanno in una di queste. È presente, oltre ai file utente, un file json che funge da “database” per gli eventi (separato dalla collection dei file degli utenti). I file utente hanno 3 campi : username, password, tickets (lista dei biglietti). Il sito mette a disposizione un API con la quale sono state eseguite le seguenti operazioni (tutte su file privati, settando quindi il campo richiesto “secret-key”, tramite setRequestProperty):

* Creazione file : è possibile farla tramite il metodo POST. E’ stata eseguita nella parte di registrazione per creare il file dell’utente. Bisogna settare anche il campo “Content-type” con “application-json” (tramite setRequestProperty). e, nel caso in cui si voglia metterlo in una collezione (come in questo caso, visto che i file utente sono raccolti in una collection), anche il campo “collection-id” con l’ID della collezione in cui vogliamo salvare il file.
* Scaricamento file : è possibile farla tramite il metodo GET. È stata eseguita sia nella parte di registrazione per scaricare la raccolta dei file degli utenti già registrati per controllare che il nome utente e la password inseriti non esistano già, sia nella parte di login per controllare che l’utente abbia scritto correttamente username e password, sia per scaricare il file json degli eventi, sia per scaricare il file utente quando devo aggiornare la sua lista dei ticket.
* Aggiornamento file : è possibile farla tramite il metodo PUT. Viene eseguita quando abbiamo prenotato dei biglietti e dobbiamo aggiornare la lista presente nel file utente sul sito. Bisogna settare anche il campo “Content-type”, come nella creazione file.

**INTERFACCIA**

La prima schermata ha 2 bottoni, in cui il primo (“registration”) consente di registrarsi e il secondo (“login”) consente di effettuare il login (vi è in questa schermata anche un menù contenente il pulsante “about”, che serve per visualizzare in un’altra schermata delle informazioni riguardanti l’app). La schermata che si apre dopo aver effettuato il login ha 4 bottoni :

1) Il primo (in alto a sinistra), “search by location”, consente di cercare eventi fornendo il nome di una località, e fa apparire una schermata in cui bisogna scrivere tale nome. Se all’interno del database vi sono eventi in quella località, verranno mostrati attraverso una lista. Cliccando su uno di questi item, verrà mostrata un’altra schermata contenente informazioni aggiuntive sull’item in questione, oltre al bottone che permette di prenotare dei biglietti. Se clicchiamo su quest’ultimo, apparirà un dialog in cui andiamo a scegliere il numero di biglietti che vogliamo prenotare. Fatto ciò basta premere sul bottone sottostante e l’operazione verrà effettuata. Si torna poi alla schermata dove sono presenti i 4 bottoni.

2) Il secondo (in alto a destra), “search by map”, consente di cercare tramite disegno su google maps. Dopo aver premuto apparirà una schermata contenente una google map e un pulsante “draw”. Quando l’utente preme sul pulsante draw, è abilitato a tracciare i contorni della zona nella quale ricercare eventi. Una volta che alza il dito dallo schermo, se ci sono eventi, essi verranno mostrati nella mappa. Cliccando su uno di essi comparirà la stessa schermata di informazioni del bottone precedente.

3) Il terzo (a sinistra, sotto il primo bottone), “view all events”, mostra una lista con tutti gli eventi presenti nel database.

4) Il quarto (a destra, sotto il secondo bottone), “tickets purchased”, fa vedere la lista dei ticket prenotati dall’utente. Un item di questa lista ha come campi il nome dell’evento e l’ID del biglietto. Accanto a ciascuno di questi biglietti vi è un bottone che serve per trasferire quello specifico ticket via NFC con un altro dispositivo (si aprirà un’ apposita schermata). Il dispositivo ricevente il ticket dovrà avere come schermata quella contenente la lista dei ticket. Una volta messi in comunicazione, gli apparirà una schermata contenente il nome dell’evento e il ticket ID inviato, con un bottone che chiede conferma per salvarlo all’interno della lista dei ticket del dispositivo ricevente. Inoltre, tenendo premuto su un ticket, sarà possibile eliminarlo (apparirà un dialog di conferma dell’azione).

**IMPLEMENTAZIONE**

Tutti i bottoni presenti nell’app sono Floating Action Button, ed ognuno ha all’interno l’icona corrispondente all’azione che svolgono (prese da Material.io). Hanno inoltre un textview sotto ciascuno che ne specifica ulteriormente l’operazione. I layout utilizzati sono di 4 tipi : coordinator layout (per le snackbar), relative layout (per le liste), uno swipe refresh layout con all’interno un constraint layout (presente nella schermata dopo il login, dove si aggiorna il database, per poter effettuare nuovamente l’operazione di aggiornamento se qualcosa va storto), constraint layout (il resto delle schermate). Le connessioni avvengono mediante la classe HttpUrlConnection. Il database interno all’app è implementato con un Room Database, che ha 2 tabelle al suo interno : EVENTS e USER\_TICKETS. La prima è la tabella che definisce i vari campi di un evento, la seconda è la tabella che ha come campi lo username e la lista di ticket associata ad esso. La Google Map è stata realizzata con un fragment e una view per permettere all’utente di disegnare la zona sulla mappa. Gli eventi eventualmente trovati sono mostrati attraverso dei marker sulla loro posizione; cliccando su uno di essi apparirà la schermata di informazioni. La prenotazione dei biglietti avviene tramite un DialogFragment, che appare dopo aver premuto il bottone di prenotazione.

Oltre alle classi che implementano le schermate (scritte nel grafo sottostante accanto alle rispettive), abbiamo le classi per la gestione del database (MyDB, che contiene il singleton del database; EventDao, che contiene le query; Converters, al cui interno abbiamo un “convertitore” per mostrare sotto forma di stringa un ArrayList, e le classi che implementano le entities del database, ovvero Event e user\_tickets) e la classe SingletonUsernameAndTicketID contenente 3 campi che sono lo username, la lista dei ticket prenotati e l’ID del file dell’utente. Sono state inoltre utilizzate alcune animazioni per rendere più gradevoli le transizioni tra una schermata e l’altra (presenta un’animazione anche la scritta “find the best events around the world”, presente nella prima schermata). Le operazioni di registrazione, login, aggiornamento dei file degli utenti e prenotazione biglietti hanno, nelle loro rispettive schermate, una progressBar che mi indica il progresso delle operazioni. Infine, ogni schermata possiede una toolbar con una freccia in alto a destra che consente di tornare alla schermata precedente.

