RELAZIONE PROGETTO FINALE:

APP PER IL CALCOLO DEL CODICE FISCALE

Giovanni Pica 0253606

Jacopo Onorati 0258245

Introduzione

Il progetto di tesi è consistito nello sviluppo di un'applicazione Android dedicata al calcolo del codice fiscale, oltreché al salvataggio in un database (in base alla volontà dell'utente) e alla visualizzazione del barcode. Le specifiche hardware minime per il corretto funzionamento è di avere un dispositivo android con Android 5.0 (o superiore).

1 Main Activity

Questa Activity è quella principale in cui l'utente dovrà inserire e quindi riempire tutti i campi riguardanti le sue informazioni personali. Tra tali informazioni abbiamo:

- Nome;
- Cognome;
- Sesso;
- Luogo Di Nascita;
- Data Di Nascita.

Per contenere tutti questi dati sono state utilizzate delle EditText per i Nomi e Cognomi, in quanto si doveva soltanto prendere in input ciò che l'utente ha inserito, per il sesso è stato utilizzato un RadioGroup, per la Data Di Nascita è stato utilizzato un DatePicker, in cui l'utente dovrà scrollare giorno mese e anno per selezionare la sua data. Invece per quanto riguarda il Luogo Di Nascita è stata utilizzata una AutoCompleteTextView, poichè bisognava suggerire all'utente il comune da inserire, infatti appena l'utente inserisce due lettere, compare una lista di comuni con quelle determinate lettere. Per fare ciò è stato utilizzato un ArrayAdapter riempito con i comuni che sono a loro volta all'interno di un database.

Dopo aver riempito tutti i campi, per calcolare il codice basterà premere il tasto “Calcola”. Si sono gestiti i casi in cui l’utente si scordi di qualche campo oppure che inserisca un comune non presente nel database, dunque sono state implementate delle exceptions:

- ComuneNonValidoException: comune non presente nel database;
- ComuneNonInseritoException: comune non inserito;
- NomeNonInseritoException: nome non inserito;
- CognomeNonInseritoException: cognome non inserito;
- SessoNonInserito: sesso non inserito.

Inoltre in questa MainActivity si ha un menù in alto che permetterà all’utente di aprire un’altra activity in cui sono salvati i codici o di toccare sui 3 puntini, per avere informazioni riguardanti la app e nostre referenze. Per implementare queste due funzionalità è stato utilizzato un menu di tipo opzioni/barra delle azioni.

2 Database Comuni

Come accennato al paragrafo precedente è stato utilizzato un database per i comuni facendo uso di Room. Per utilizzare Room bisognava prima di tutto inserirlo nel build.gradle (Module:app), una volta fatto questa sono state create:

- DatabaseComuni: la classe del database di comuni, è stato utilizzato inoltre il pattern Singleton per impedire l’apertura simultanea di più istanze del database;
- Comune: classe entity della tabella comune, che ha come attributi nome e codice catastale, il nome è una stringa concatenata con nome comune e la provincia di appartenenza tra parentesi;
- ComuneDAO: interfaccia dao che contiene i metodi per accedere nel database, tra cui:
    - List<Comune> getAll() che ritorna una lista di comuni;

    - String[] loadNames() che ritorna un’array di nomi;
    - String getCode(String nomCom) che ritorna il codice catastale inerente a quel nomCom;
    - String getName(String nomCom) che ritorna il nome di quel comune, questo metodo serve per controllare se un certo comune è presente nel database;
    - void insertAll(List<Comune> comuni) serve per inserire una lista di comuni nel database.

Per inserire la lista di comuni nel database, si è utilizzato un file csv contenente un elenco aggiornato di comuni. Per leggere tale file è stata implementata una funzione readCsvFile() in cui si salvano nome, sigla automobilistica (provincia) e codice catastale, che corrispondono rispettivamente alla sesta, tredicesima e diciottesima colonna della tabella del file csv.

3 Calcolo

In questo paragrafo si incomincerà a parlare del calcolo vero e proprio del codice fiscale. Prima di tutto è stata creata una classe CalcolaCF, che viene istanziata per la prima volta nella MainActivity ed avrà come attributi:

- Nome;
- Cognome;
- Sesso;
- Codice Catastale;
- Giorno Di Nascita;
- Mese Di Nascita;
- Anno Di Nascita.

Inoltre essa ha altre quattro variabili di tipo String che vengono popolate dalle vocali/consonanti del nome e cognome. Per semplificare la gestione di caratteri che sono consonanti o vocali si è creata una classe Parole, che controlla se un carattere è una vocale ed

inoltre ha dei metodi che servono per trovare stringhe di caratteri in posizione pari e dispari che ci serviranno per il calcolo del carattere di controllo. Una volta istanziata tale classe, la MainActivity chiama il metodo di calcolo di essa e quindi verranno calcolati i vari codici:

- Codice Cognome: sono le prime tre consonanti, ma se il cognome ha meno di 3 consonanti verranno inserite le prime vocali, se ancora abbiamo poche vocali ne cognome andremo ad aggiungere delle “X” a seconda di quante lettere ci mancano;
- Codice Nome: sono la prima, terza e quarta consonante, come per il cognome, se abbiamo meno di 3 consonanti dobbiamo aggiungere le prime vocali e se il numero di vocali non basta, allora bisogna aggiungere delle “X”;
- Codice Data Di Nascita e Sesso: a seconda dell’anno si prendono le ultime due cifre, a seconda del numero del mese avremo una lettera corrispondente ed infine a seconda del giorno e del sesso, se il sesso è maschile allora il giorno rimane così com'è, altrimenti si aggiunge 40 al giorno;
- Codice Catastale: si prende dal database a seconda del comune di appartenenza;
- Carattere Di Controllo: ha funzione di controllo dell'esatta trascrizione dei primi quindici caratteri. Esso viene determinato nel modo seguente: ciascuno degli anzidetti quindici caratteri, a seconda che occupi posizione di ordine pari o posizione di ordine dispari, viene convertito in un valore numerico in base alle corrispondenze indicate rispettivamente ai successivi punti 1) e 2).

  1) Per la conversione dei sette caratteri con posizione di ordine pari:

  A o zero = zero O = 14
  B o 1 = 1 P = 15

C o 2 = 2	Q = 16
D o 3 = 3	R = 17
E o 4 = 4	S = 18
F o 5 = 5	T = 19
G o 6 = 6	U = 20
H o 7 = 7	V = 21
I o 8 = 8	W = 22
J o 9 = 9	X = 23
K = 10	Y = 24
L = 11	Z = 25
M = 12	- -
N = 13	- -

2) Per la conversione degli otto caratteri con posizione di ordine dispari:

A o zero = 1	O = 11
B o 1 = 0	P = 3
C o 2 = 5	Q = 6
D o 3 = 7	R = 8
E o 4 = 9	S = 12
F o 5 = 13	T = 14
G o 6 = 15	U = 16
H o 7 = 17	V = 10
I o 8 = 19	W = 22
J o 9 = 21	X = 25
K = 2	Y = 24
L = 4	Z = 23
M = 18	- -
N = 20	- -

I valori numerici determinati vengono addizionati e la somma si divide per il numero 26. Il carattere di controllo si ottiene convertendo il resto di tale divisione nel carattere alfabetico ad esso corrispondente nella sottoindicata tabella:

zero = A	14 = O
1 = B	15 = P
2 = C	16 = Q
3 = D	17 = R
4 = E	18 = S
5 = F	19 = T
6 = G	20 = U
7 = H	21 = V
8 = I	22 = W
9 = J	23 = X
10 = K	24 = Y
11 = L	25 = Z
12 = M	- -
13 = N	- -.

4 Fragment

Una volta premuto il tasto calcola, si aprirà un DialogFragment, che si comporterà come se fosse una nuova finestra, ma minimizzata rispetto ad una activity. In questo fragment avremo informazioni inerenti all'utente che ha fatto richiesta del calcolo tra cui:

- Nome;
- Cognome;
- Data Di Nascita;
- Codice Fiscale;

- Barcode associato.

Per il calcolo del barcode (Code 39) sono stati utilizzati delle funzionalità della libreria standalone ZXing, che prima del suo effettivo utilizzo è stata inserita nei build.gradle (Module:app).
La funzione che lo calcola è la generateCode(String value, ImageView iv) che genererà un codice sotto forma di bitmap, per poi utilizzare una setImageBitmap() con la ImageView passata in input, che conterrà appunto il barcode.
Tutte queste informazioni sono state inglobate all'interno di una ImageView sotto forma di Tessera Sanitaria. Per fare ciò si è utilizzato un LinearLayout, perchè appunto permetteva di scrivere all'interno di una ImageView. Inoltre al di sotto di questa tessera, sono stati inseriti due bottoni, per far scegliere all'utente di salvare o meno nel database di persone il codice appena richiesto.

5 Database persone

Anche per implementare il database delle persone è stato utilizzato Room. Il database delle persone viene chiamato quando l' utente decide di salvare il codice Fiscale calcolato .
Le classi utilizzate per implementare il database sono:

- PersonaDatabase è stato utilizzato inoltre il pattern Singleton per impedire l'apertura simultanea di più istanze del database;
- Persona che crea delle entity aventi come attributo un nome, un cognome, il codice fiscale, la data di nascita e un id;
- PersonaDAO che effettua le query necessarie alle varie activity.

6 Favourites

L' activity Favourites è quella che si occupa di implementare una RecyclerView per mostrare all' utente la lista dei "Preferiti", cioè degli utenti salvati in memoria . Favourites fa in modo di mostrarvi

una semplice lista delle persone con il loro nome e cognome. Per fare ciò il metodo `createPerson()` compie un' interrogazione del database ottenendo tramite `loadCodes()` una lista di tutti i codici fiscali salvati e tramite `getName()` e `getLastName()` i nome e cognomi. Per ogni nome e cognome viene creata un unica stringa nome-cognome. Quest'ultima viene passata al metodo giveMePerson della classe Reduced_Person, la quale crea un oggetto Reduced_Person avente solo codice e nome-cognome come attributi, createPerson() ci darà alla fine una lista di oggetti Reduced_Person. Questa lista viene passata all'Adapter della RecyclerView, che prima di tutto deve essere importata nel file build.gradle (Module:app). L'aver creato un oggetto di tipo Reduced_Person è stato fatto partendo dal concetto che in possibili futuri interventi sull' app la lista dei Favoriti rimarrebbe comunque sempre costituita dai soli nomi e cognomi, oltre alla comodità del poter creare righe composte da un'unica TextView. Quando l'utente clicca una riga viene fatta un Intent che lancia l' activity Detail alla quale viene passato l'oggetto Reduced_Person.
Da segnalare c'è anche l'uso di una classe SetAlternateCol. Un oggetto di tipo SetAlternateCol viene istanziato nella onBindViewHolder, metodo dell' Adapter della RecyclerView. Il metodo `setColor()` di SetAlternateCol fa in modo da settare le righe pari della RecyclerView di un colore diverso da quello del layout xml che si occupa della grafica della singola riga.

7 Detail

L' activity Detail mostra all'utente il dettaglio sull'oggetto che ha salvato, in particolare il nome e cognome, il codice fiscale, la Tessera Sanitaria e il Barcode calcolato.
Detail prende tramite il bundle passato dall' Intent lanciata da Favourites l'oggetto Reduced_Person selezionato dall'utente e tramite

`giveMeCode()` e `giveMeName ()` ,metodi di Reduced_Person ottiene nome e cognome e il codice fiscale dell'oggetto scelto dall'utente per mostrarli dell TextView del layout relativo.
Inoltre Detail implementa un menu per effettuare la cancellazione di un oggetto dal database . Cliccando sul cestino, l'app mostra un DialogFragment che chiede conferma della volontà di cancellare l'elemento dal salvato dal database. Il fragment viene lanciato in `onOptionsItemSelected()` passandogli un bundle contenente il solito oggetto di tipo Reduced_Person..

8 Fragment per la cancellazione

Il fragment lanciato dall' activity Detail chiede conferma della cancellazione dell'oggetto scelto dal database. Esso è implementato dalla classe FragmentDeletion. Nella `onCreateView()` viene preso l'oggetto di tipo Reduced_Person dal bundle e ne vengono ricavati il nome e cognome, per mostrarli nella conferma di cancellazione e il codice fiscale che serve per effettuare la query sul database per ottenere l'oggetto Persona nel caso l'utente spinga "Si". La query è data dal metodo di personaDAO `getPerson()` che restituisce un oggetto persona.