UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Tesi di laurea Triennale in

Informatica

IMPLEMENTAZIONE DI UN'APP ANDROID PER RACCOMANDARE RICETTE CULINARIE

Relatore

Prof. Vincenzo Deufemia

Candidato

Giulio Di Maria
Matr.: 05121 03518

Anno Accademico 2017/2018

Ringraziamenti

Grazie. Grazie all'università con tutti i suoi docenti e collaboratori per aver rappresentato ciò di cui avevo bisogno, non solo come studente ma come persona. Una crescita personale e professionale nella quale il vero risultato non è stato quello degli esami ma l'esperienza di vita. Grazie ai miei colleghi che hanno condiviso con me stress, speranze, momenti di gioia, e dure ore di lavoro. Ho collaborato con persone che non conoscevo e che poi sono diventate miei amici. Scambiandoci appunti dei corsi, libri di testo, spiegazioni, parlando di problemi personali e soprattutto sapendo che se si ha una difficoltà in un qualsiasi campo, si poteva contare uno sull'altro, e sono sicuro che tutto questo resterà impresso nella mia mente finché ne potrò aver memoria.

Un ringraziamento particolare alla mia famiglia, che da vicino o da lontano mi ha supportato per gli insuccessi e festeggiato nelle vittorie.

Indice dei contenuti

Indice delle figure

CAPITOLO 1
INTRODUZIONE

Siamo ormai in un'epoca in cui il "machine learning" (apprendimento automatico) ricopre un ruolo di rilievo nella vita di tutti i giorni. Lo studio dei "big data" associato ad algoritmi sempre più performanti ed efficienti sono riusciti a digitalizzare ogni informazione in modo tale da adattarsi alle varie situazioni e necessità. Buona parte di questi dati viene acquisita durante l'utilizzo di applicazioni tramite l'inserimento, a volte involontario ma sempre consenziente, di informazioni da parte di utenti di tutto il mondo. Secondo il sito statista[1], infatti, sono circa 65 miliardi le applicazioni scaricate dal Google Play app store durante solo il mese di maggio 2016. Le somme di denaro e gli interessi socio-economici sono quindi rilevanti.

In questo documento di tesi viene presentata un'applicazione sviluppata per dispositivi Android che attraverso un algoritmo di raccomandazione basato su filtering collaborativo e di contenuti suggerisce all'utente ricette culinarie d'interesse.

[1] https://www.statista.com/statistics/281106/number-of-android-app-downloads-from-google-play/

Le ricette sono state ricavate dal sito Giallo Zafferano tramite un software di web scraping (harvesting). I dati sono stati salvati come records in tabelle di un database relazionale (usando MySQL) e come file all'interno di un sistema gerarchico di cartelle con minore granularità al fine di creare una risorsa di backup. I dati salvati nel database sono stati esportati in ulteriori cartelle all'interno del progetto Android, tramite un altro programma. Questa scelta puramente diretta all'efficienza funzionale del sistema, garantisce una reperibilità locale di tutti i dati e riduce sensibilmente la tempistica di installazione del software sul device a scapito della memoria. Tramite il settaggio per uso a lungo termine dei gusti dell'utente (rispetto agli ingredienti) e delle sue allergie, l'algoritmo di raccomandazione effettua una prima selezione delle ricette che possano piacere. Un ulteriore filtro è aggiunto dal settaggio dei parametri a breve termine che sono: il tempo a disposizione per cucinare, il costo che si vuole sostenere per la preparazione della pietanza, le abilità culinarie e la tipologia della ricetta (se si vuole un primo, un secondo, etc..). Infine dopo un adeguato utilizzo dell'applicazione da parte di più utenti si può affinare ancora di più l'algoritmo di raccomandazione, permettendo agli utenti dai gusti simili di consigliarsi le pietanze in base alla loro esperienza. Questo ultimo passaggio viene effettuato tramite l'analisi dello storico delle ricette gradite o meno, il coefficiente di correlazione di Pearson e una media tra il consiglio ottenuto da utenti simili e la media dei voti che ogni ricetta consigliata già possiede.

1.1 Struttura della tesi

Il lavoro di tesi di laurea presenterà la seguente struttura generale:

- Ringraziamenti
- Abstract
- Indice dei contenuti
- Indice delle figure
- Indice delle tabelle
- Capitolo 1 Introduzione
- Capitolo 2 Recipe Advisor, un'applicazione per raccomandare ricette
- Capitolo 3 Costruzione della base di dati
- Capitolo 4 Algoritmo di raccomandazione
- Capitolo 5 Interfaccia grafica utilizzata
- Capitolo 6 Conclusioni
- Riferimenti Bibliografici
- Appendice A

Il capitolo 1 presenta un riassunto breve e conciso degli argomenti trattati. In sinergia con l'abstract fornisce una visione strutturale generale del progetto e di ciò che quest'ultimo intende evidenziare.

Il capitolo 2 precorre i principi che saranno trattati nei capitoli 3-5. Viene quindi spiegata la logica di business instradando il lettore attraverso le varie fasi di sviluppo dell'applicazione, dall'origine della base di dati fino al meccanismo di generazione dell'algoritmo, e quindi del prodotto finito.

Il capitolo 3 caratterizza il software di web scraping. Le finezze di come è stata estrapolata ogni informazione utile dalle pagine web di Giallo Zafferano e di come queste siano state memorizzate all'interno del sistema. Si affronteranno i dettagli della struttura relazione adoperata, della gerarchia di back-up realizzata ad hoc, e della conversione necessaria per la migrazione dei record dal server locale di Mysql al modulo dedicato dell'applicazione Android.

Nel capitolo 4 si espongono gli elementi alla base dell'algoritmo di raccomandazione e di come quest'ultimo riesca a garantire un esito positivo degli obiettivi prefissati. Un'analisi più approfondita dei singoli elementi matematici impiegati viene fornita all'appendice A.

Il capitolo 5 descrive l'interfaccia utente in modo schematico e modulare. Alcuni delle entità grafiche adottate derivano dallo studio della letteratura di Head First Android Development citato nella bibliografia.

Nell'ultimo capitolo ogni unità del complesso progettuale è menzionata figurativamente al fine di dimostrare esaustivamente il raggiungimento degli obiettivi preposti. A coronamento del lavoro svolto, viene esposto un possibile sviluppo della tesi successivo al corso di studi.

Nell'appendice A viene esposto un approfondimento degli argomenti statistici e probabilistici a base dell'algoritmo.

Capitolo 2
RECIPE ADVISOR,
un'applicazione per raccomandare ricette

Ogni realizzazione ben organizzata ed efficiente si erige su una struttura robusta e consistente. Ogni modulo, infatti, deve essere funzionalmente valido per il corretto esercizio del sistema. Per l'ideazione dell'architettura ho cercato di soddisfare tali requisiti e di evitare l'aggiunta inutile di componenti non funzionali al fine di ottimizzare sia l'utilizzo della memoria che delle prestazioni dell'applicazione.

In questo capitolo verrà analizzata la suddivisione dei vari elementi all'interno del progetto da un punto di vista generale e rappresentativo. Seguendo il cammino dei dati all'interno dell'architettura sarà possibile scoprire ogni servizio offerto e le sue caratteristiche, così da comprenderne le motivazioni per cui sia stato ideato. Al termine del capitolo sarà chiaro ogni collegamento tra i vari componenti e le loro funzionalità all'interno del contesto.

2.1 Organizzazione della logica di business

In seguito ad un attento studio in fase di progettazione sono riuscito a realizzare lo schema sottostante:

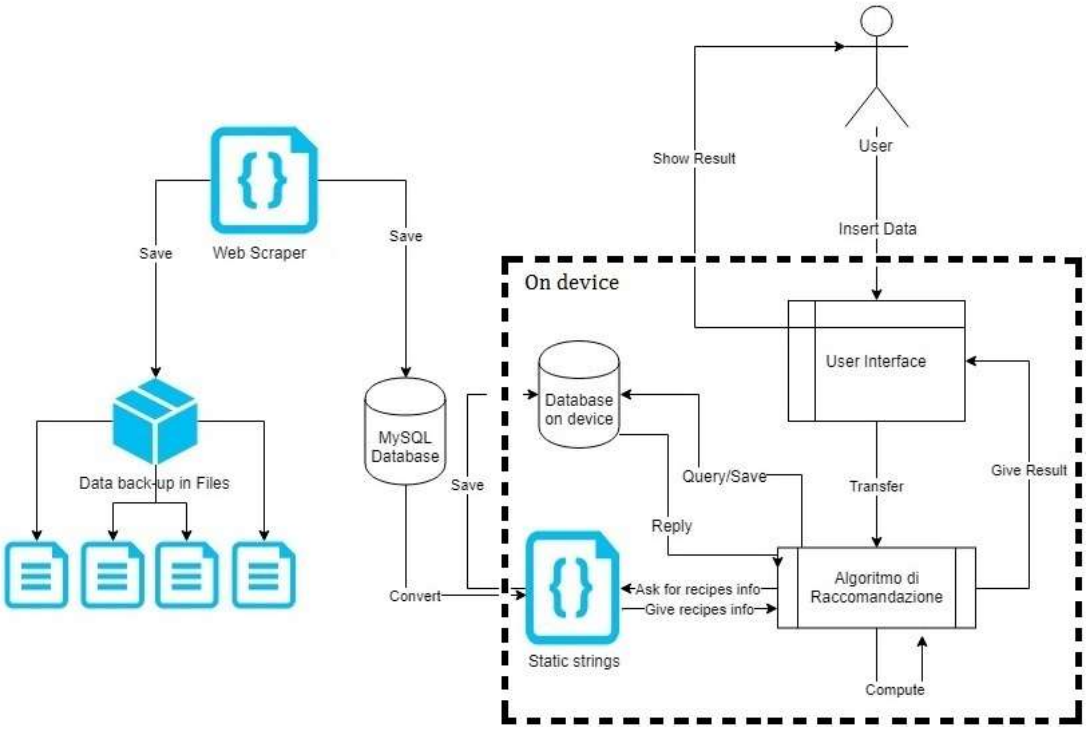

FIGURA 2.0.1 SCHEMA DELLE RISORSE UTILIZZATE ALL'INTERNO DEL PROGETTO

Come si può notare immediatamente osservando la Figura 2.0.1, l'applicazione non è gestita interamente all'interno del dispositivo mobile, ma parte dello sviluppo è avvenuto all'esterno di essa, infatti, l'elemento fondamentale che ha fornito tutte le informazioni per il popolamento del database è stato il software di web scraping da me ideato.

La scelta di creare un algoritmo di questo tipo al di fuori del prodotto finale, per poi esportare i dati necessari per quest'ultimo durante una fase successiva dello sviluppo, è dettata da ragioni di efficienza. In effetti essendo che un dispositivo mobile gode di una riduzione delle prestazioni dovuta alla gestione in economia della batteri , e di una capacità di calcolo di gran lunga inferiore a quella di un elaboratore fisso, mi è sembrato obbligatorio considerare di non appesantire troppo l'inizializzazione del prodotto che sarà già inficiata dall'installazione del database locale. Inoltre volevo che l'app funzionasse completamente offline in modo che potesse essere sempre operativa anche in mancanza di una connessione di rete. Logicamente se avessi voluto inserire del codice che apre un flusso di dati per recuperare stringhe da una pagina web, l'applicazione non sarebbe mai potuta restare locale.

Il web scraper non gode di molta elasticità essendo stato realizzato con l'intento specifico di riuscire a estrapolare le informazioni unicamente dal sito di Giallo Zafferano. Per riuscire nel mio intento ho ricercato l'indirizzo della pagina del sito che contenesse una lista con tutti gli elementi. In seguito tramite il metodo openStream della classe URL ho aperto una connessione verso l'indirizzo contenente tale indice e da lì ho aperto una nuova connessione verso ogni linea di codice che contenesse il titolo di una ricetta. Per identificare che l'indirizzo conducesse alle pagine da me desiderate ho considerato solo gli url che fossero contenuti in una classe con tag header 2 (h2) e dal nome title-recipe. Con metodi simili a quello appena presentato, ho sfruttato le caratteristiche della programmazione in html per poter recuperare, da ogni connessione aperta, tutti i dati che potessero risultare utili ai fini dell'applicazione.

Così come sono state scaricate le stringhe, allo stesso modo, sono state scaricate tutte le immagini delle ricette ottenute. Una volta scaricati questi contenuti, sono stati memorizzati all'interno della base di dati su MySQL.

Come già descritto, il software che esegue il download dei dati è impostato ad hoc per il contesto in cui è servito. Questo significa che nel caso lo sviluppatore decidesse di apportare delle modifiche al codice che si trova sulle pagine online, oppure chiudesse definitivamente il sito, non avremmo più la possibilità di accedere alle informazioni. Ecco perché oltre a creare il database relazionale ho pensato di creare una gerarchia di cartelle in modo da avere una risorsa di back-up in caso di necessità. Così nel caso ci sia un malfunzionamento nel database, oppure i dati venissero involontariamente danneggiati durante la loro manipolazione, si avrà sempre una garanzia permanente a disposizione da cui estrapolare tutto ciò di cui si può aver bisogno, senza dover considerare lo stato del sito online.

Nella rappresentazione della Figura 2.0.1, ho omesso il programma che fa da intermediario tra il database che fornisce i dati e il model che ne fa la richiesta. Questa omissione è del tutto voluta per evitare di prendere troppo spazio con informazioni comuni e banali. Indico comunque che per la connessione al database in questa parte del progetto è stata utilizzata l'API JDBC (Java Database Connectivity), per ulteriori informazioni a riguardo è possibile consultare la pagina aggiunta nelle note.[2]

[2] https://www.oracle.com/technetwork/java/javase/jdbc/index.html

Una volta completata la sezione riguardante tutto ciò che andasse fatto al di fuori dell'applicazione mobile, ho iniziato a curarmi di come trasferire i risultati ottenuti nell'ambiente interno.

In base alle accurate ricerche che ho portato a termine, non esiste una metodologia standard che importi un database da MySQL ad un progetto Android senza l'utilizzo di una connessione ad internet. Quindi ho deciso ancora una volta di sviluppare un software apposito in Java che mi permettesse di ottenere i risultati desiderati. Non potendo spostare direttamente il database da una parte all'altra, ho deciso di creare delle classi final che contenessero tutte le stringhe con le informazioni utili.

Per realizzare questo obbiettivo, ho modificato il metodo toString delle classi che avevano oggetti all'interno del database in quel momento (Ingredients e Recipes). Dopodiché ho creato un nuovo metodo main che prendesse tutti i records dalla base di dati, e per blocchi di duecento elementi, creasse stringhe costanti numerate contenenti ciò che volevo spostare. All'inizio di ogni classe che è stata creata ho inserito un costrutto condizionale switch che mi permettesse di selezionare con facilità le stringhe d'interesse in base al file e allo spiazzamento all'interno del file modulo 200.

Così facendo ho ottenuto i dati all'interno dell'ambiente di sviluppo mobile senza ricorrere a memorie in remoto o software di terzi.

Essendo l'applicazione basata sull'inserimento dei gusti personali dell'utente, ho dovuto implementare un'interfaccia che agevolasse il più possibile queste iterazioni. Ho mantenuto un profilo di semplicità e chiarezza nella presentazione dei contenuti interattivi, aggiungendo grafiche e testi intuitivi e preservando l'esperienza dell'utente

indirizzata all'impiego del tempo nello scoprire nuove ricette, invece che alla ricerca di una guida per l'utilizzo di un'interfaccia troppo complessa.

Logicamente l'obbiettivo principale si concentra nel fornire un algoritmo che possa consigliare la ricetta con l'indice di gradimento più alto possibile. Per fare questo, ho preferito ottimizzare le prestazioni non facendo sempre ricorso al database, ma facendo richiamare al codice sorgente dell'algoritmo le classi statiche precedentemente create per il passaggio dei dati. Così facendo i tempi si riducono vertiginosamente. Infatti, mentre con il database bisognava fare circa 4000 query per recuperare tutte le ricette e per ognuna di queste in media altre 7 query per recuperarne tutti gli ingredienti utilizzati, tramite l'invocazione diretta delle classi statiche è bastata fare una sola invocazione. Per riuscire in questo intento ho salvato tutte le stringhe in un unico oggetto Json (JavaScript Object Notation) e le ho inviate tramite la serializzazione in una cartella privata dell'app (sfruttando le Shared Preferences fornite dal sistema). Per utilizzare oggetti Json in Android ho importato una libreria esterna trovata su GitHub chiamata Gson.[3]

Purtroppo non tutte le informazioni potevano essere trasferite in questo modo. Infatti vi sono alcune tabelle più dinamiche (come quelle che contengo le allergie, le pietanze piaciute e non da parte dell'utente) che sono più piccole in dimensioni rispetto a quelle delle ricette e potrebbero quindi far sprecate più tempo durante la serializzazione e deserializzazione che nel produrre delle query mirate al database in Android.

[3] https://github.com/google/gson

Allora dopo varie valutazioni e test, ho optato per un sistema misto composto da un array di ricette serializzato (con all'interno anche gli ingredienti di ogni ricetta) e da tabelle di piccole dimensioni contenente i settaggi personali degli utenti. Nel caso in cui l'applicazione potesse richiedere un maggior quantitativo di memoria, allora si potrà provvedere a passare anche queste tabelle al processo di serializzazione tramite l'utilizzo di matrici.

Alla fine del processo, quando l'algoritmo di raccomandazione termina di elaborare i dati, il risultato prodotto viene dinamicamente caricato all'interno di un layout per la presentazione all'utente.

L'utente prendendo visione del risultato, può continuare a interagire all'interno dell'app modificando i dati, lasciando recensioni delle ricette di cui ha preso visione (tramite query al database), oppure ricercandone di nuove modificando i parametri a suo piacimento.

CAPITOLO 3

COSTRUZIONE DELLA BASE DI DATI

In questo capitolo si analizzerà la struttura e la composizione del database nel dettaglio, con una particolare attenzione verso quella tipologia di dati di cui si è considerata importante l'utilità a lungo termine, e per cui sono state create delle tabelle in modo da memorizzarli permanentemente.

Le informazioni più volatili non sono state considerate all'interno del database, ma al termine del loro utilizzo, le locazioni di memoria utilizzate da questi elementi vengono liberate da parte del Garbage Collector, il quale ricopre un ruolo importante tanto in Android quanto nel mondo di Java in generale.

Alla fine di questa sezione sarà possibile avere una chiara visione di come è stata costruita la base di dati e di quali siano stati, invece, gli elementi che non hanno ricevuto una memorizzazione permanente. Questa distinzione ci aiuterà a comprendere di quali dati l'algoritmo avrà a disposizione ogni volta che gli verrà richiesto di raccomandare una ricetta all'utente, e di questi, quali resteranno per la prossima richiesta.

3.1 Configurazione tabellare

Prima di creare effettivamente una base di dati bisogna comprendere quali siano le necessità delle informazioni di cui si ha bisogno. Ho cercato di comprendere quali potessero essere gli elementi sfruttando le metodologie apprese durante il corso di studi. Tramite uno studio minuzioso, ho adoperato meccanismi di ereditarietà tra le classi ritenute utili nella fase di analisi dei requisiti, creando un diagramma ER e studiando le varie relazioni con cardinalità, generalizzazioni, specializzazioni, mapping e reificando lì dove necessario.

Al termine di tutte queste operazioni e di una valutazione del consumo di memoria e della capacità di calcolo, di tutte le queries che potessero derivare dalla ristrutturazione dello schema ER, ho costruito il database così come presentato nella Figura 3.0.1.

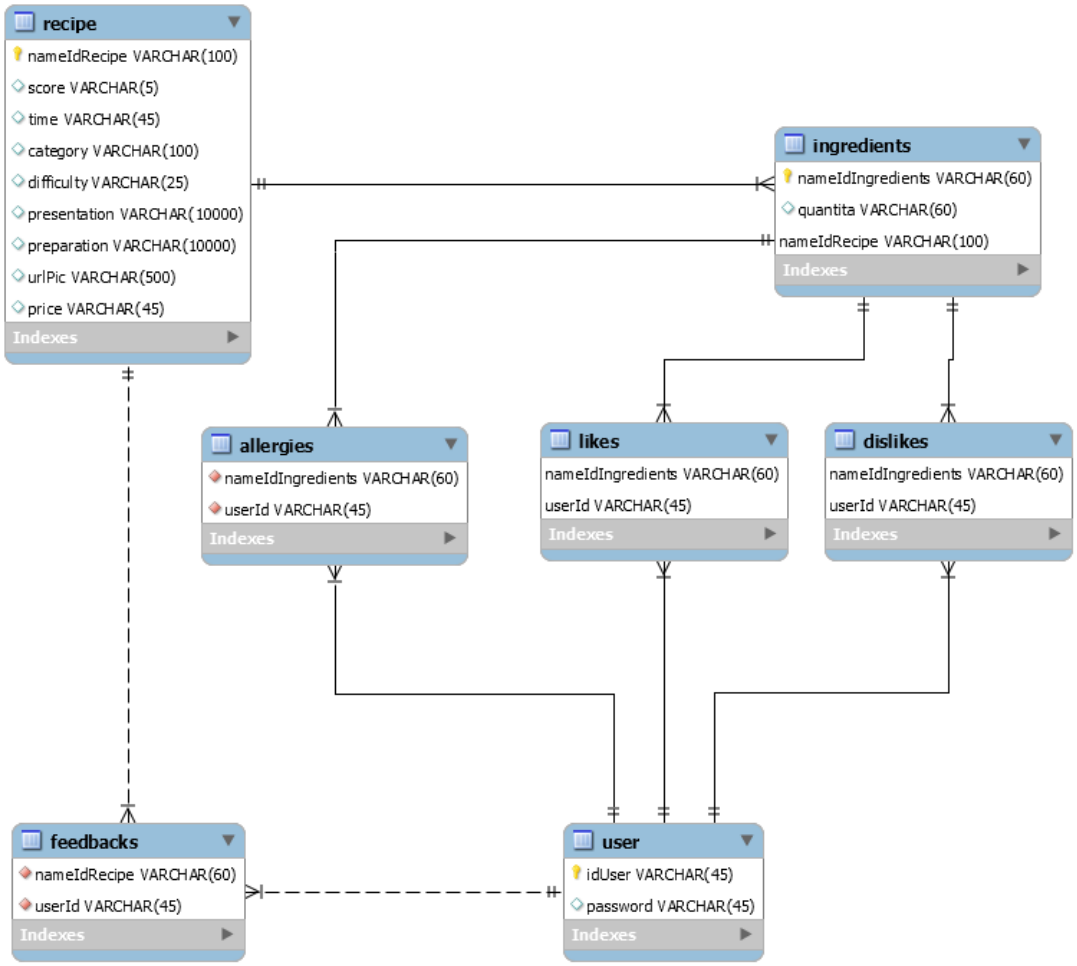

FIGURA 3.0.1 SCHEMA EER DELLA STRUTTURA DEL DATABASE SU MYSQL E SULL'APPLICAZIONE ANDROID

Come si può notare, non vi sono percorsi troppo dispendiosi da una classe all'altra. La struttura realizzata collega in modo efficiente ogni tabella alle altre in modo che recuperare un'informazione non comporti un eccessivo spreco di risorse. Si può osservare anche che i componenti più importanti sono rappresentati dall'utente e dagli ingredienti. Queste due classi non sono casuali. Infatti, da un punto di vista generale ogni interrogazione che può essere utile al fine della raccomandazione si può ricollegare ad una di queste di loro due.

3.2 Dati gestiti in modo permanente

Procediamo analizzando ogni singola classe. La tabella Recipe, nucleo fondamentale nella logica di business, contiene tutte le ricette collegate all'applicazione. Ogni ricetta logicamente deve contenere intrinsecamente gli ingredienti ad essa collegati. Per questo possiamo avere che per ogni ricetta siano collegati più ingredienti, ed ogni ingrediente può essere collegato a più ricette. Al termine della popolazione del database, tramite queries appositamente costruite, sono riuscito a rilevare che ci sono ingredienti presenti in più di 1800 ricette (ad esempio il sale fino), e ricette che posseggono fino a 22 ingredienti (come lo Zelten, pane dolce fruttato tradizionale del Sud-Tirolo). Altra caratteristica della tabella Recipe è lo score. Questo attributo deriva direttamente dai contenuti scaricati dal software di web-scraping del capitolo 2, e rappresenta un valor medio delle recensioni ottenute dal sito Giallo Zafferano relativamente ad ogni ricetta. Questa informazione viene largamente sfruttata sia per il cold start (fase iniziale di utilizzo del software in cui l'approccio collaborativo è ancora impossibile per via della poca conoscenza degli utilizzatori), che per affinamenti successivi dell'algoritmo di raccomandazione.

La tabella degli ingredienti si identifica tramite il nome dell'ingrediente e l'identificativo della ricetta al quale l'ingrediente è associato. Questo utilizzo della chiave esterna è fondamentale. Se non venisse utilizzata, e si identificassero gli ingredienti solamente in base alla loro chiave primaria, in presenza di ingredienti contenuti in più ricette avremmo l'interruzione dell'applicazione seguita dal lancio di

un'eccezione riguardante la definizione della stessa chiave primaria in molteplici records (la classe che fornisce l'eccezione ed andrebbe quindi gestita è SQLException per il connettore Connector/J, quest'ultimo fornisce i driver necessari per gestire un database MySQL in Java).

Vi sono poi quelle classi che si ottengono dalle relazioni "molti a molti" tra l'utente e gli ingredienti. Queste classi reificate e costruite settando le chiavi esterne degli oggetti in cui sono in relazione come chiavi primarie, gestiscono le informazioni necessarie per il filtraggio delle ricette nell'algoritmo di raccomandazione. Ogni utente al momento dell'iscrizione imposta dei parametri per riferire i propri gusti e le proprie allergie. Questi sono dati da mantenere in modo permanente perché se non tenessimo conto ad esempio che un determinato profilo è allergico ad un ingrediente in particolare, potremmo consigliare una ricetta che non è adatta all'utente che sta utilizzando l'applicazione. Inoltre i contenuti riguardanti i gusti personali aiutano a filtrare le ricette selezionando e modificando il valore di gradimento di quest'ultime. Senza un valore permanente ci ritroveremo a richiedere un ulteriore settaggio ad ogni richiesta di raccomandazione (probabilmente ogni settaggio sarebbe uguale a quello che lo precede, essendo che i gusti personali e le allergie di un individuo difficilmente si modificano, relativamente a tempistiche adeguate).

La tabella dell'username contiene tutti i dati relativi a colui che interagisce con l'interfaccia grafica. Qui salviamo le informazioni d'accesso, ovvero l'identificativo dell'utente (username) e la password per accedere al proprio account.

In ultimo, ma non per importanza, possiamo osservare la tabella dei feedbacks. Oltre a contenere il nome della ricetta per cui si rilascia il feedback e l'utente che ha rilasciato tale riscontro, nella base di dati in Android è stata aggiunta una votazione della ricetta da 1 a 5. La tabella in sé, rappresenta le visualizzazioni da parte degli

utenti delle schede contenenti le ricette. L'aggiunta de valore assegnato, attributo addizionale presente nel database in Android, invece, è molto utile per avere un mezzo di comparazione tra i vari utenti e comprendere quanto due di quest'ultimi possano essere simili tra loro. Infatti, come vedremo nella spiegazione dell'algoritmo, ogni feedback assegnato, viene utilizzato dal sistema per comparare quanto siano affini due profili e metterli eventualmente in correlazione per affinare i suggerimenti delle ricette.

Vi sono dati che non hanno bisogno di essere permanenti e che quindi vengono settati tutte le volte che si vuole richiedere una raccomandazione. Il nome che definisce tali dati è collegato alla memoria principale che li gestisce (la Ram), detta anche memoria "volatile".

3.3 Dati di natura volatile

Nella vita di tutti i giorni ci sono fattori in continua mutazione. Se pensiamo a quando dobbiamo preparare una pietanza, valutiamo sempre la tempistica necessaria per la preparazione. Questo parametro non è sempre lo stesso, ma dipende da vari fattori (come gli impegni che si presentano prima e dopo la preparazione della pietanza, o il desiderio del tempo massimo che si vuole spendere in cucina, etc..).

A meno che non si vogliano sviluppare dei dati statistici a riguardo, conservare in memoria secondaria un parametro permanente per un'informazione di questo tipo sarebbe un grande spreco di spazio e di capacità di calcolo. Andremmo ad effettuare delle query di modifica a ciascuna raccomandazione senza poter utilizzare questa informazione in futuro e non calcolando che le queries di questo tipo potrebbero essere di un numero molto elevato.

Quindi ho deciso per motivi prestazionali e logici, di non memorizzare la tempistica di preparazione come un parametro definitivo, ma di mantenerlo volatile. Oltre alla tempistica anche il costo che si vuole sostenere per la preparazione della pietanza, il livello di difficoltà della ricetta e la tipologia (primi piatti, secondi piatti, contorni, antipasti o dolci) sono temporanei.

L'unico elemento che potesse creare dei dubbi sulla necessità di memorizzazione permanente è la difficoltà della ricetta. Ho considerato l'idea di lasciar settare in modo definitivo le proprie capacità culinarie. Ho anche compreso però che spesso ci si vuole mettere alla prova, magari provando a cucinare qualcosa che vada al di là delle nostre reali capacità.

Inoltre, a volte capita di non essere da soli in cucina, ma di avere il sostegno di un cuoco più esperto ed a proprio agio con determinate complessità di preparazione. Per queste ragioni, le abilità in cucina vengono inserite tra i settaggi a breve durata.

Abbiamo così presentato l'origine degli elementi non facenti parte della base di dati come extra, all'interno di un contesto dedicato, invece, ai dati destinati alla base di dati.

Nonostante la tesi possa variare con l'upgrade verso nuove versioni in seguito a modificazioni post-laurea, a prova del buon lavoro svolto, ritengo che la struttura presentata in questo capitolo, per la sua coerenza ed efficienza, rimarrà invariata o al limite potrà essere utilizzata come base per ulteriori integrazioni tabellari e quindi relazionali.

CAPITOLO 4
ALGORITMO DI RACCOMANDAZIONE

Considerando l'esposizione della struttura dei dati del capitolo precedente, siamo ora in grado di poterci concentrare sulle caratteristiche computazionali dell'algoritmo.

Verranno rappresentate le due tipologie di filtraggio utilizzate al fine di valutare le varie caratteristiche di ogni ricetta per comprenderne la validità alla luce della conoscenza dell'utente. La prima basata sui contenuti, i quali saranno scansionati singolarmente in modo esclusivo e qualitativo. La seconda basata sulla collaborazione e dei principi matematici ad esso collegato.

Infine, andremo a considerare il risultato finale, e di come questo venga semplicemente realizzato valorizzando al meglio il lavoro svolto durante la fase di filtraggio.

4.1 Il processo di Filtering

Considerando la vasta gamma di scelta possibile, e considerando che si desidera proporre un'unica ricetta, risulta il bisogno indiscutibile di selezionare le migliaia di ricette in modo che venga fornita solo la migliore tra tutte.

FIGURA 4.0.1 GERARCHIA DI FILTERING UTILIZZATA

In effetti per impostare una corretta selezione dobbiamo sfruttare al massimo proprio quelle conoscenze che abbiamo acquisito in input e che sono descritte nel capitolo 3. Ogni singola informazione contiene degli elementi che possono far prevalere una ricetta rispetto ad un'altra.

Come gestire quindi la mole di dati e le varie informazioni in modo che sinergicamente forniscano il risultato voluto? L'idea è di dividere per categoria di filtraggio le varie caratteristiche acquisite, come mostrato nella Figura 4.0.1.

Ad esempio, un'occorrenza la cui presenza è sicuramente indesiderata, tra le ricette consigliate, è un piatto contenente un ingrediente a cui l'utente ha dichiarato di essere allergico, pertanto, la caratteristica delle allergie deve essere inserita all'interno di una famiglia di filtri che selezioni le ricette in base al contenuto, in modo esclusivo. Un altro esempio potrebbe essere l'ingrediente che non piace molto all'utilizzatore dell'applicazione. Nel caso avessimo un elemento sgradito, di sicuro dobbiamo tenerne conto, ma questa peculiarità non determina l'esclusione della ricetta, essendo che nonostante tutto la ricetta possa comunque rientrare tra le migliori scelte che si hanno a disposizione.

Andiamo quindi ad analizzare le varie tipologie di filtraggio utilizzate e di come queste vengano applicate all'interno dell'algoritmo di raccomandazione.

4.1.1 Filtering Selettivo

Le informazioni che vogliamo utilizzare per scartare alcune delle ricette o tutte dal numero totale considerato in input appartengono alle seguenti tipologie:

- Allergie
- Tempistiche
- Difficoltà di preparazione
- Costo
- Categoria

Considerando gli elementi già presentati nel precedente capitolo, si consideri che un ulteriore classe all'interno del progetto per l'app provveda a recuperare i dati forniti dal database in MySQL ed importarli correttamente (la classe utilizzata si chiama MyDatabaseHelper, l'ho realizzata estendendo SQLiteOpenHelper, che a sua volta è parte del package android.database.sqlite).

Riteniamo che l'algoritmo abbia già recuperato tutte le informazioni presenti nella base di dati e si prepari a computarle.

Le allergie settate della tabella utente sono in relazione con gli ingredienti. È su quest'ultimi che faremo la nostra selezione, essendo essi collegati anche ad ogni ricetta. Effettuando una scansione di tutte le pietanze e di tutte le allergie, ogni volta

che troviamo almeno un ingrediente in comune tra i due insiemi, la ricetta presa in considerazione viene scartata.

Per impostazioni come il tempo massimo che si intende trascorrere nella preparazione della pietanza, il costo massimo che si intende sostenere e il livello di complessità culinario che si è in grado di sostenere il modus operandi è elementare. Semplicemente viene assegnato un valore crescente partendo dal valore minimo, fino a quello massimo. Vengono analizzate tutte le ricette, e ogni qual volta si presentasse una ricetta che non rispetti uno di questi tre parametri, quest'ultima viene scartata.

Discorso simile ma leggermente discostante può essere concepito per la categoria. Quest'ultima selezione segue un criterio che è molto simile a quello utilizzata per le allergie, solo che non vengono scartate tutte le ricette che contengono la categoria selezionata, ma al contrario, vengono scartate tutte le pietanze le cui categorie non siano uguali a quella selezionata.

Se non rimane nessuna ricetta valida, viene visualizzato un messaggio a video che segnala l'accaduto e consiglia di modificare i parametri.

Terminata questa fase, si procede al filtering sui contenuti qualitativo, portando avanti nell'algoritmo i risultati raggiunti.

4.1.2 Filtering Qualitativo

Procuratosi le ricette valide relativamente ai parametri inseriti dall'utente, si ricerca come affinare ancora di più la selezione.

Per riuscire in questo intento, senza contare il filtering collaborativo, si possono solo utilizzare le informazioni rimanenti per favorire determinare ricette caratterizzate da particolari elementi (questa sezione fa ancora parte del cold start).

Proprio per questo ideale si analizzano le seguenti informazioni:

- Likes
- Dislikes
- Rating già acquisiti
- Ricette già consultate

Nel contesto progettuale, con likes intendo tutti quei ingredienti che l'utente ha indicato di apprezzare particolarmente. Con dislikes, invece, tutti gli ingredienti che l'utente ha indicato di non apprezzare.

Seconda una personale analisi ho considerato che degli ingredienti con una denotazione negativa sfavoriscano maggiormente una determinata ricetta rispetto a quanto ingredienti con una denotazione positiva la possano avvantaggiare. Pensiamo, ad esempio ad un gelato. Se su un cono gelato ci sono tre ingredienti che non ci piacciono e due che invece apprezziamo, considereremo il gelato come non gradito. Anche se ci fossero solo due gusti su cinque che non apprezziamo comunque il gelato potrebbe non piacerci. Seguendo questo ragionamento (estratto pragmaticamente

dall'osservazione di diversi soggetti che mi hanno fornito il loro feedback riguardo l'assaggio di alcune pietanze), ho ottenuto che per l'algoritmo, ogni due dislikes equivalgono tre likes. Quindi: $1 Dislike = 1{,}5\ likes$. Non potendo però valutare quanto di ogni ingrediente ci fosse all'interno della ricetta (essendo che non tutti utilizzano delle unità di misura standard, ma alcune volte il quantitativo è rappresentato da "quanto basta" o "il giusto"), così da fornire una giusta assegnazione e valore al like o al dislike, ho normalizzato il numero di likes e dislikes in base al numero totale di ingredienti nella ricetta.

Infatti, anche se non posso dimostrare matematicamente che un ingrediente è più presente in una ricetta invece che in un'altra valutandone effettivamente la quantità, posso sicuramente affermare che normalmente un ingrediente presente in una ricetta di tre ingredienti è sicuramente più importante dello stesso ingrediente in una ricetta che di ingredienti ne ha venti.

Il valore massimo che gli elementi nei likes possono aggiungere al risultato già ottenuto di una determinata ricetta è di 5 punti; mentre il valore massimo che gli elementi nei dislikes possono sottrarre al risultato già ottenuto di una determinata ricetta è di -7,5 punti (comunque non passando mai il limite dei zeri punti con un andamento negativo).

Con "risultato già ottenuto" che ho citato nel paragrafo immediatamente precedente a questo, intendo il valore dei feedback che la ricetta ha ottenuto su Giallo Zafferano al momento in cui l'ho estrapolato. Come già detto quel valore è compreso tra 1 e 5. Quindi il valore di partenza di una ricetta va da 1 a 5 punti e alla fine del filtering qualitativo può essere compreso tra 0 e 10 punti.

Il filtering sulle ricette già consultate non modifica l'intervallo appena presentato. Infatti per evitare che venga riproposta sempre la stessa ricetta che è valutata come la

più gradita, si decrementa il risultato delle pietanze a scaglioni del 15%, 30% e 45% in base al fatto che siano state consultate già 1, 2, o 3 e più volte.

4.1.3 Filtering Collaborativo

Il valore reale ottenuto varia quindi da 0 a 10. Abbiamo terminato l'analisi di tutti gli elementi forniti dall'utente. Come possiamo ancora migliorare il filtraggio in modo da fornire un risultato che sia ancora più affidabile?

Sfortunatamente non si è ancora in grado di fornire un'intelligenza artificiale pensante che sia capace di consigliare all'utente sfruttando le sue opinioni personali, fortunatamente però, altri utenti che hanno utilizzato l'applicazione avranno avuto sicuramente un'opinione di ciò che hanno già visionato. Quella opinione viene registrata come feedback all'interno del database che è stato costruito. Quindi potremmo teoricamente confrontare quale utente abbia avuto i gusti più simili ai nostri e farci consigliare da quest'ultimo una nuova ricetta da assaggiare.

Ci si potrebbe chiedere se sceglierne uno sia giusto oppure risulti troppo restrittivo. Se considerassimo che ad ogni raccomandazione, potrebbe essere cambiato utente che consiglia la ricetta, e che in oltre non dipende tutto da lui, il risultato proposto non sembra così tanto fuori luogo. Difatti, anche l'utente che ci consiglia la ricetta potrebbe averla ricevuta da qualcuno con gusti simili ai suoi (quindi abbastanza simili anche ai nostri) che lo ha portato ad apprezzare un determinato prodotto. Quindi non staremo seguendo l'opinione di un solo utilizzatore, ma bensì di due. Continuando questo ragionamento per migliaia di persone e considerando che la raccomandazione da parte degli utenti simili non esclude la valutazione effettuata con gli altri elementi di filtering (più tardi valuteremo il perché di questa affermazione), scopriamo così che la rete di raccomandazioni che si crea è abbastanza fitta da garantire un risultato di qualità.

Ma come è possibile comprendere quali utenti abbiano gusti affini ai nostri? Un' idea potrebbe essere di analizzare gli ingredienti che piacciono sia a lui che a me, ma questo approccio sarebbe di poco spessore perché modificherebbe solo leggermente la valutazione fornita dal filtering selettivo e qualitativo. Quindi un'altra soluzione possibile più performante potrebbe essere di analizzare i feedback lasciati alle varie ricette di cui gli altri utenti hanno preso visione e confrontare se hanno rilasciato feedback "linearmente simili ai nostri". Ciò che si fa è proprio valutare su uno stesso numero di ricette di cui entrambi gli utenti hanno preso visione, il numero di feedback assegnati dall'utente che ha richiesto la raccomandazione e quelli presentati dagli altri. Se si verifica un forte parallelismo nella linearità dei punti su di un piano, allora questi due utenti presentano una similarità tra loro.

Uno strumento molto utile per analizzare la similarità in questo modo sfrutta l'indice di correlazione di Pearson[4]. L'immagine presentata è una rappresentazione grafica dei possibili valori assunti dal coefficiente. Tale figura appartenente alla pagina di Wikipedia citata nelle note.

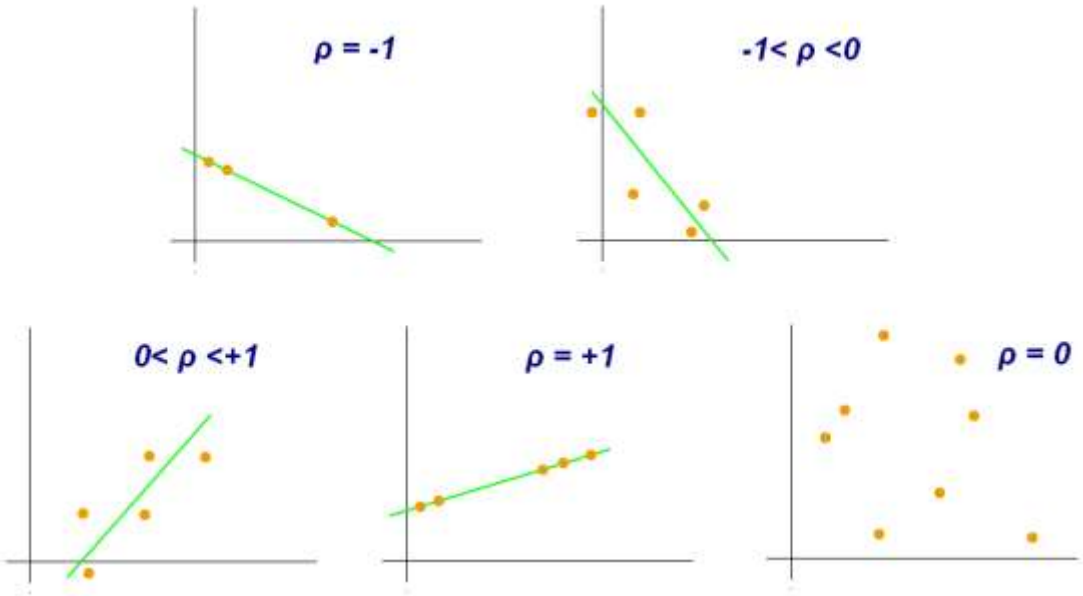

FIGURA 4.3 VALORI CHE PUÒ ASSUEMERE L'INDICE DI CORRELAZIONE DI PEARSON

La formula del coefficiente di Pearson è la seguente: $\rho_{AB} = \frac{cov(A,B)}{\sigma_A * \sigma_B}$

Dove cov(A,B) rappresenta la covarianza di A e B, mentre il simbolo greco sigma minuscolo in posizione interna rappresenta la deviazione standard campionaria di A e B.

[4] https://it.wikipedia.org/wiki/Indice_di_correlazione_di_Pearson

Ulteriori informazioni riguardanti la covarianza, la deviazione standard campionaria ed il perché il loro rapporto in questa formula è sempre compreso tra -1 ed 1 sono approfondite nell'appendice A di questo lavoro di tesi.

Ottenuto un valore che definisce quanto siano in relazione due utenti, possiamo considerare quanto un utente sia affidabile nel dare raccomandazioni in base a quanto sia simile a colui a cui sono destinate. Va notato (ed è facilmente visibile in Figura 4.0.13) che sia essa una correlazione fortemente positiva o negativa, ovvero che tende molto verso gli estremi dell'intervallo, avremmo in ogni caso due utenti molto simili tra loro.

Allora, per migliorare ancora di più l'algoritmo presentato fino alla fine degli altri due filtraggi, si classifica l'utente che abbia un coefficiente più alto possibile, considerando il valore assoluto derivante dalla funzione. Una volta scoperto l'elemento che è più indicato a dare raccomandazioni, si valuta la ricetta che lui abbia più gradito e si aggiunge un valore pari al risultato ottenuto dai filtraggi precedenti, per il valore del coefficiente per due.

Ecco la rappresentazione grafica della formula:

$$New\ score = old\ score * reliabilty * 2$$

Dove new score rappresenta il nuovo punteggio che la ricetta otterrà. Old score rappresenta il punteggio ottenuto grazie ai due filtering precedentemente descritti e reliability è l'indice di correlazione che figurativamente rappresenta l'affidabilità dell'utente nel consigliare.

Questa raccomandazione aggiunge valore solo alle ricette che, l'utente a cui bisogna suggerire, non ha ancora provato. Per cui il valore massimo che si può ottenere in aggiunta, grazie al filtering collaborativo, a quello che una pietanza già possiede è di 10 punti. Dovendo essere il punteggio già ottenuto dalla ricetta sul sito

di Giallo Zafferano uguale a 5, più i 5 punti per i likes, non vi deve essere nessun decremento, e i due utenti devono aver dato esattamente le stesse recensioni. Otteniamo quindi come risultato alla formula 10*1*2 = 20. Dove 5 sono del vecchio punteggio, 5 ottenuti con i likes e 10 con la raccomandazione.

Dunque alla fine di tutti i filtraggi, il risultato che può ottenere una determinata pietanza contenuta nel database è compresa tra 0 e 20. Ricordiamo comunque che se un utente è stato ritenuto simile, ma ci consiglia una ricetta che non ci piace o comunque non ci rende più d'accordo, e quindi assegniamo un valore divergente da quello che ha dato lui, alla prossima iterazione probabilmente verrà selezionato un altro utente per la raccomandazione.

Ulteriori sviluppi dell'algoritmo potrebbero portare ad implementare un sistema che tenga presente i primi tre utenti più simili e un valore normalizzato per la gestione di punti extra da aggiungere alle ricette. Quindi non più solamente alla miglior selezionata da parte del suggeritore, ma a tutte quelle che ha considerato positivamente.

Alla fine di tutto questo lavoro di selezione, le ricette restanti con il relativo punteggio acquisito vengono messe a confronto. La ricetta con il punteggio più alto sarà l'output fornito dall'algoritmo.

CAPITOLO 5
INTERFACCIA GRAFICA UTILIZZATA

Per rendere fruibile l'applicazione c'è bisogno che questa sia intuitivamente utilizzabile. Empiricamente si sa che l'utente medio non ama dover passare molto tempo nel comprendere come poter utilizzare un determinato software, specialmente se quest'ultimo non concerne motivazioni lavorative. Si preferisce sempre un prodotto che si possa riuscire ad utilizzare sfruttando l'istinto.

Per questo, tramite le conoscenze acquisite durante il corso di studi, ho analizzato ogni layout in modo da renderlo più semplice e chiaro possibile, considerando un utente con basilari conoscenze di utilizzo dei dispositivi mobili e condizioni di accessibilità nella norma.

Nelle prossime pagine sarà presentato il lavoro di coordinamento dei vari layout in base ai principi che sono stati appena esposti.

5.1 Schema generale

L'interfaccia grafica che è stata costruita mira a fornire gli elementi necessari per il funzionamento dei moduli presentati durante i capitoli precedenti.

Nulla è stato lasciato al caso, anche per l'implementazione del front-end è stato speso un certo quantitativo di energie al fine di riuscire tramite la ricerca e l'apprendimento ad ampliare le proprie conoscenze nella composizione del layout e dell'interazione uomo-macchina.

La rappresentazione grafica infatti non è delle più basilari ed affronta alcuni elementi non banali. A tal proposito ritengo che sia stato davvero molto utile lo studio per l'approfondimento affrontato con la letteratura presentata nella bibliografia.

In questa sezione si analizzeranno le idee che si celano dietro ogni implementazione (grafica) di ogni interfaccia fornita all'interno del progetto. Partendo dalla Home Page (main activity, logicamente specificata nel file xml manifest come launcher), fino al layout che rappresenta il risultato derivante dal lavoro dell'algoritmo di raccomandazione.

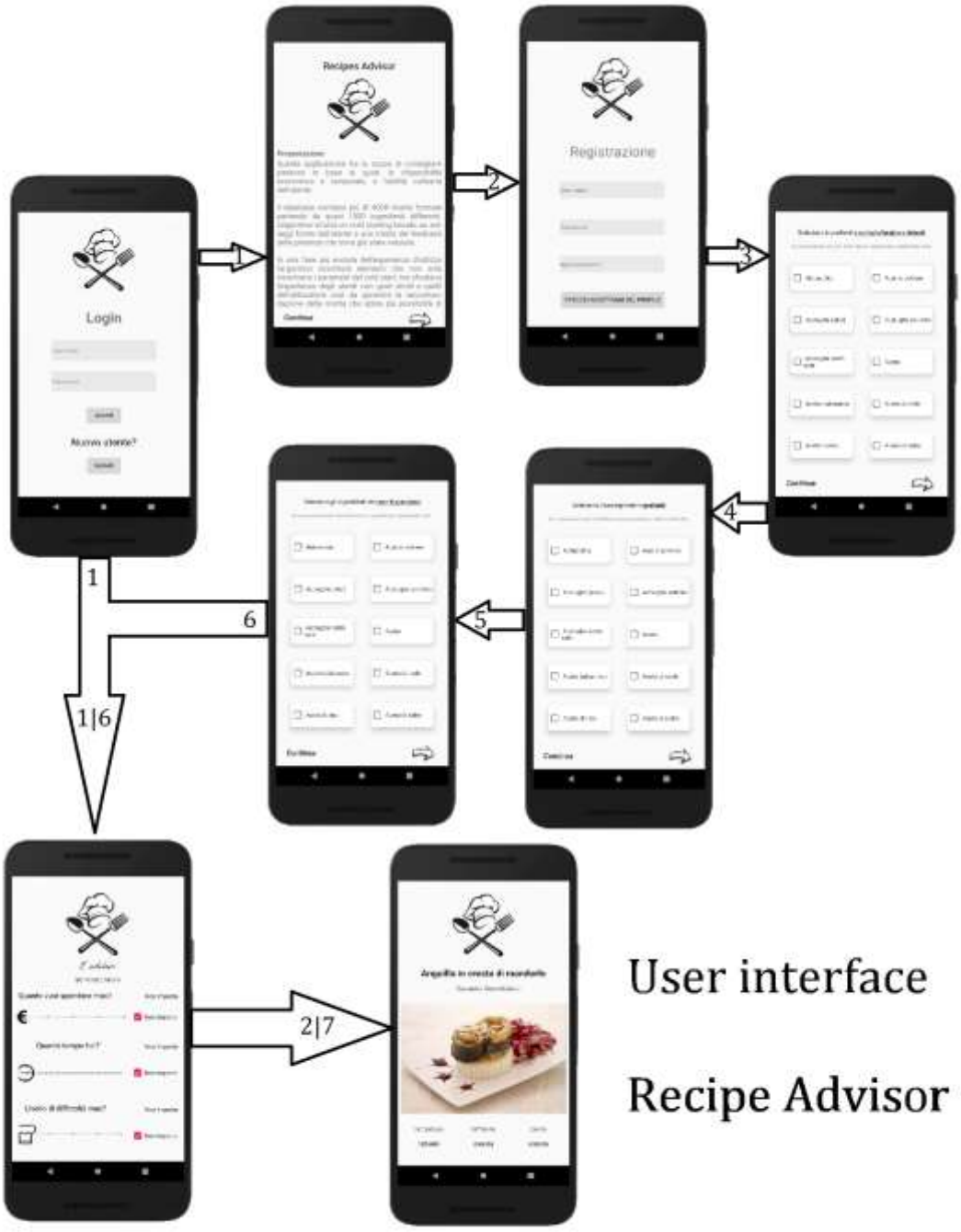

FIGURA 5.0.1 TUTTI I POSSIBILI PERCORSI DELL'INTERFACCIA ALL'INTERNO DEL PROGETTO

L'utilizzo dell'applicazione è molto rapido ed efficiente. Solo al primo accesso viene richiesto di effettuare un'iscrizione che impegni del tempo supplementare a causa dei settaggi obbligatori da effettuare.

Si possono distinguere quindi due percorsi possibili all'interno dell'interfaccia, uno utilizzato dall'utente che deve registrarsi ed esegue l'accesso per la prima volta, ed un altro che viene eseguito dall'utente già registrato.

Come si può notare dallo schema presentato in Figura 5.0.1, il tragitto percorso da parte dell'utente già registrato, che dall'home page segue le frecce 1 e 2 fino al risultato, viene ricoperto anche dall'utente in fase di registrazione, che invece segue le frecce da 1 a 5 e da 6 a 7 le condivide con gli utenti già registrati essendo anche lui diventato uno di loro. Quindi per praticità proseguirò nel descrivere l'interfaccia partendo dall'utente in fase di iscrizione e terminando con l'utente registrato che richiede una raccomandazione.

5.2 Iscrizione

Per i riferimenti riguardo ciò che verrà presentato in questa sezione e nella successiva rapportarsi alle indicazioni fornite nella Figura 5.0.1.

Al momento dell'avvio dell'app l'utente non registrato non potendo accedere all'account è obbligato a crearne uno eseguendo una pressione del bottone su cui è scritto "iscriviti" (passando così sulla freccia 1 dell'immagine). Nella nuova schermata che compare viene presentata un'introduzione alle funzionalità dell'applicazione. Essendo la presentazione abbastanza lunga è possibile da visualizzare solo una porzione per volta, e per consultare la parte nascosta basta scorrere con il dito il testo verso l'alto. Alla fine dell'introduzione viene indicato che si può procedere al prossimo layout sia premendo sulla freccia in basso a destra, sia facendo click su continua (passando sulla freccia numero 2 dell'immagine).

Nella terza schermata vengono presentati tre rettangoli dove è possibile inserire del testo e in ognuno di essi, prima che questi vengano riempiti, è suggerito l'input che andrebbe inserito. Vengono effettuati controlli sull'effettivo inserimento di un username valido, ovvero che non sia stato già utilizzato da parte di altri utenti, e sulla egualità della password inserita in entrambe le caselle ad essa dedicate. Nel caso non venissero passate le verifiche descritte, un Toast comparirà nella parte inferiore dello schermo descrivendo quale dei vincoli necessari per una corretta registrazione non è stato correttamente soddisfatto. Se tutti i criteri vengono effettivamente superati con

esito positivo, l'utente passa alla fase di settaggio dei parametri conoscitivi dell'utente (passando così sulla freccia 3 dell'immagine).

In questo layout la cosa più immediata da notare sono le opzioni per la selezione dei vari ingredienti presentate con uno stile in rilievo e mantenenti lo stesso stile grafico degli altri elementi presentati finora. Questa particolare lista di elementi non è una presentazione standard che viene utilizzata nei manuali base front-end per Android, bensì ricorre ad un Recycler view che permette di ottimizzare i consumi di memoria per lunghe liste di elementi e che tramite un custom adapter [5] fornisce i contenuti da inserire all'interno di cardViews sostenute da ViewHolders. I cardViews sono elementi che spesso si notano in applicazioni come quella del Google Play store dei dispositivi Android, in cui è presente un'immagine e delle scritte all'interno di un rettangolo. Questo tipo di interfaccia è una versione più avanzata e flessibile del normale listView che invece è contenuto all'interno delle classi fornite in modo standard dalla piattaforma Android.

Possiamo notare come in questo layout si sia mantenuta la freccia in basso a destra uguale a quella utilizzata nella pagina di introduzione, così da dare la possibilità all'utilizzatore di non dover ricercare o riflettere troppo su come dover procedere alla fase successiva.

Tornando indietro da questo punto non si ritorna alla schermata precedente, ma viene visualizzato un messaggio che informa della possibilità di perdere i dati inseriti. In caso di due click successivi si ritorna alla Home Page sprecando i progressi raggiunti.

[5] Un adapter agisce figurativamente come un ponte tra i dati e l'elemento grafico che vuole utilizzarli. Un custom adapter è una caratterizzazione di questo elemento che nel mio caso non prende neanche spunto dalle classi standard fornite da Android, ma è stata creata ad hoc per il problema in questione.

Tutte le frecce che portano dalla grafica appena descritta fino alla fase finale di registrazione, mantengono sempre la stessa presentazione (frecce 4 e 5). Dall'ultima schermata però, invece di procedere verso un altro settaggio vengono salvati tutti i progressi effettuati (username, password, allergie, likes e dislikes settati) e si presenta all'utente la stessa schermata che viene fornita agli utenti registrati e che hanno effettuato il login (freccia 6).

5.3 Login

Dopo essersi registrato o aver effettuato il Login l'utente arriva alla schermata rappresentata in figura con la freccia 1|6. Questa schermata dà la possibilità di modificare i settaggi temporanei che vengono richiesti ad ogni raccomandazione. L'interfaccia anche qui è molto intuitiva e ben coordinata tra i vari elementi che la compongono. Le slider inserite, contengono thumbs con immagini che ricordano l'operazione che stiamo impostando. Al moviemento del thumb si cambia l'impostazione desiderata e la modifica è immediatamente visibile all'interno del testo in alto a destra. Ogni impostazione ha collegata ad essa una textView ed un bottone "Non importa" che permette di non settare alcun elemento per quel determinato parametro.

Una volta che tutti i dati sono stati impostati all'interno di questo layout si procede tramite la pressione del bottone "Consigliami", questo pulsante viene gestito da un

listener che lo ascolta e ne esegue il relativo codice alla pressione, dando il via all'algoritmo che fornirà la ricetta che ha più possibilità di poter piacere all'utente.

Nel caso invece si premesse il pulsante indietro da questo punto, si riceverebbe un messaggio di allerta che informa dell'imminente logout dal sistema (ritornando alla Home Page) e chiede se si è sicuri di voler procedere. In caso si prema ok, oppure ancora indietro, il sistema riporterà alla pagina di Home con le credenziali da inserire oppure l'iscrizione da effettuare. In caso di rifiuto si ritorna alla schermata così com'era presentata prima della pressione del pulsante indietro.

Il risultato fornito in output dall'algoritmo viene inviato all'ultima activity per mostrare ciò che si è ottenuto (freccia 2|7). Quest'ultima interfaccia da una visione chiara e organizzata della ricetta. Si compone, in effetti, di una grande distribuzione di informazioni ben descritta da titoli chiaramente collegati ai relativi contenuti. In fondo alla pagina si trova una rateBar che permette di registrare un feedback riguardo alla pietanza che è stata suggerita.

Per uscire da questa schermata basta premere indietro, e si ritorna al layout contenente i settaggi. Da qui sarà possibile richiedere un ulteriore raccomandazione o effettuare il logout.

CAPITOLO 6
CONCLUSIONI

Come è verificabile da una lettura e analisi del lavoro svolto, gli obiettivi presentati all'inizio del progetto sono stati ampiamente raggiunti. Riuscire a realizzare in ogni sua parte ciò che qualche mese fa rappresentava solo un'idea nella mia testa mi ha dato grande soddisfazione.

Per uno sviluppo futuro vorrei aggiungere nuovi task ed apportare migliorie grafiche ed algoritmiche al fine di pubblicarne il risultato. In particolare desidererei integrare il lavoro sviluppato con i nuovi frigoriferi smart, così da poter raccomandare le ricette non solo con i filtraggi presentati, ma in base agli ingredienti disponibili. Il servizio che ne deriverà non solo porterà un prodotto finito sul mercato, ma potrà significativamente migliorare le mie capacità da programmatore.

Riferimenti bibliografici

[1] Dawn Griffiths, O'Reilly Media book, *Head First Android Development 2nd edition,* *https://dogriffiths.github.io/HeadFirstAndroid/#/*

[2] I Sistemi di raccomandazione, https://fr.slideshare.net/valeriagennari94/sistemi-di-raccomandazione

[3] Android: Handling Checkbox state in Recycler Views, https://android.jlelse.eu/android-handling-checkbox-state-in-recycler-views-71b03f237022

APPENDICE A

Questa sezione è dedicata ad un approfondimento dei concetti che hanno fornito le basi per l'utilizzo del coefficiente di correlazione di Pearson. Le conoscenze presentante in questa sezione sono state acquisite durante lo studio per l'esame di "Calcolo della probabilità e statistica matematica".

In questa particolare sezione affronteremo una piccola introduzione dedicata al valore atteso e alla definizione di varianza (informazioni necessarie per la comprensione degli argomenti successivi), per poi trattare la covarianza e la deviazione standard.

Tramite questi due ultimi elementi si potrà capire perché il rapporto nella formula del coefficiente di Pearson è sempre un valore compreso tra -1 ed 1.

A.1 Il valore atteso

Definiamo una variabile aleatoria un valore reale definito da una funzione nello spazio campionario. Quindi abbiamo un determinato valore reale (appartenenti all'insieme dei numeri reali), che è aleatorio, ovvero assume un valore diverso dipendendo da un determinato evento (ad esempio il lancio di una moneta, in cui la variabile può assumere valori 0 o 1), ed è definito da una funzione (come già detto in base ad un determinato evento il valore considerato muta, la relazione tra il valore che abbiamo prima e quello che

otteniamo in seguito all'evento rappresenta la funzione) nello spazio campionario, ovvero all'interno dell'insieme contente tutte le possibilità risultanti dall'esito della funzione.

Quando la variabile aleatoria può assumere solo un'infinità numerabile di valori è detta discreta. Quindi, per la definizione appresa in precedenza, una variabile aleatoria è discreta se e soltanto se il suo spazio campionario è finito (detto anche numerabile essendo che possiede la stessa cardinalità dei numeri naturali, ovvero, può essere messo in corrispondenza biunivoca con l'insieme dei numeri naturali. La spiegazione e dimostrazione di questo argomento non verrà ulteriormente approfondita per evitare di divagare più di quanto non stia già facendo).

Considerato un evento in x. Il valore atteso della variabile aleatoria discreta è la sommatoria di tutti i valori assumibili da x moltiplicata per il peso della loro densità discreta. In formula:

$$E(X) = \sum_{x:p(x)>0} x * p(x)$$

La densità discreta è l'insieme delle probabilità che un determinato evento possa accadere. In parole povere, il valore atteso (o speranza matematica) è la sommatoria di tutte le probabilità maggiori di zero ($x: p(x) > 0$) che ha la variabile x di accadere, moltiplicata per la probabilità che questo avvenga. Nella sua concezione più generale il valore atteso risulta banalmente essere una media pesata di una variabile aleatoria.

A.2 La varianza

Sebbene il valore atteso possa esprimere un valore significativo delle possibilità degli eventi all'interno dello spazio campionario, non ne determina la dispersione, o varianza. Se consideriamo la variabile aleatoria X con valore atteso μ, la varianza di X viene definita nel seguente modo:

$$Var(X) = E[(X - \mu)^2] = \sum_{x:p(x)>0} (x - \mu)^2 * p(x)$$

Avendo prima affermato che il valore atteso non è altro che una media pesata, è facile allora analizzare la prima parte dell'equazione. Dopo un breve ragionamento infatti si può capire che non è altro che il valore atteso della differenza della variabile aleatoria rispetto al suo valore atteso, il tutto al quadrato. Ovvero, stiamo dando valore alla distribuzione dei valori e da quanto questi si distanzino dal valore medio pesato. L'elevazione al quadrato oltre che fornire valori sempre positivi permette di rendere più significativo il risultato ottenuto. La seconda uguaglianza invece si avvicina ancora di più a quanto presentato in precedenza, infatti l'unica modifica effettuata rispetto alla formula del valore atteso è stata di aggiungere la differenza ed elevare il tutto al quadrato. Leggendo quindi la formula se ne comprende anche il significato. Ogni x ha un peso diverso. Se una x con una densità discreta che tende ad 1 si discosta molto dal valore atteso, farà aumentare di tanto la varianza ed influenzerà quindi il risultato molto di più rispetto ad una x che si discosta poco, ma ha una densità discreta inferiore.

A.3 La deviazione standard

La definizione di deviazione standard deriva intuitivamente da quella di varianza. Presentando la formula d'interesse sarà tutto immediatamente più chiaro:

$$\sigma_X = \sqrt{Var(X)} = \sqrt{E[(X - \mu)^2]}$$

Molto banalmente la deviazione standard rappresenta la varianza che viene riportata alla stessa unità di misura del valore atteso. Infatti, si può facilmente notare che varianza ha la stessa unità di misura di X^2 mentre la deviazione standard ha la stessa unità di misura di X.

A.4 La Covarianza

Riflettendo sul concetto di valore atteso potremmo analizzare la seguente formula:

$$Cov(X, Y) = E[(X - E[X])(Y - E[Y])]$$

Leggendola ci accorgiamo che non stiamo facendo altro che mettere in relazione la distribuzione dei valori di X rispetto al suo valore atteso con i valori della distribuzione di Y rispetto al suo valore atteso, essendo la moltiplicazione una operazione che gode della proprietà commutativa, anche la distribuzione di Y rispetto al suo valore atteso è messa in relazione con la distribuzione di Y rispetto al suo valore atteso, abbiamo quindi una correlazione tra X e Y ("coVARIANZA", il nome suggerisce che vi è una relazione reciproca

rispetto alla varianza). Da qui si può facilmente comprendere il significato della formula. Ovvero comprendere quanto siano simili le distribuzioni delle due variabili.

Se $Cov(X,Y) > 0$, allora le due distribuzioni si corrispondono positivamente e se una di esse tende ad assumere valori maggiori o minori alla propria media, anche l'altra variabile sosterrà lo stesso andamento.

Se $Cov(X,Y) < 0$ allora le due distribuzioni si corrispondono negativamente e le loro tendenze saranno inverse una all'altra. Mentre nel caso in cui $Cov(X,Y) = 0$ logicamente capiamo che tra i due andamenti delle due distribuzioni non esiste una correlazione valida.

A.5 La correlazione acquisisce sempre un valore compreso tra -1 e 1

Avendo analizzato i concetti precedentemente presentati, ridiamo un'occhiata al coefficiente di correlazione di Pearson:

$$\rho_{AB} = \frac{cov(A,B)}{\sigma_A * \sigma_B}$$

Seguendo quanto visto, possiamo riscrivere la formula nel seguente modo:

$$\rho_{XY} = \frac{E[(X - E[X])(Y - E[Y])]}{\sqrt{E[(X - E[X])^2] * E[(Y - E[Y])^2]}}$$

Come osservato nella sezione precedente due variabili aleatorie X e Y hanno la $Cov(X,Y) = 0$ se e solo se non sono in relazione tra loro. Quindi è facile comprendere che il coefficiente diventa pari a zero quando il numeratore diventa uguale a zero (al contrario il denominatore non diventerà uguale a zero a cause delle operazioni esponenziali).

Seguendo la logica lineare del contesto potremmo riscrivere Y come aX + b ovvero come una funzione lineare con coefficiente angolare a e termine noto b.

Apportando quindi questa modifica alla formula della covarianza avremmo:

$$Cov(X, aX + b) = E[(X - E[X])(aX + b - E[aX + b])]$$

Semplificando avremo: $E[(X - E[X])(aX + b - E[aX + b])] =$

$E\left[\sqrt{\{(X - E[X])(aX + b - E[aX + b])\}^2}\right] = E[\sqrt{\{aX + b - aE(X) - b\}^2}] =$

$E[a\sqrt{\{X - E(X)\}^2}]$ = a Var(X)

Mentre seguendo una logica simile potremmo trasformare la deviazione standard nella seguente formula:

$$\sigma_X * \sigma_{aX+b} = \sqrt{Var(X)a^2Var(X)}$$

Il termine noto si annulla durante lo sviluppo, così com'è avvenuto per la semplificazione durante la covarianza.

Il risultato finale dopo le semplificazioni risulta:

$$\rho_{x,ax+b} = \frac{a\,Var(X)}{\sqrt{Var(X) * a^2 * Var(X)}} = \frac{a}{|a|} = \begin{cases} 1, & se\ a > 0 \\ -1, & se\ a < 0 \end{cases}$$

Da questa dimostrazione abbiamo ottenuto quindi che il coefficiente di correlazione e limitato nell'intervallo 1,-1 e che nel caso valga zero, i duo valori X e Y non sono correlati fra loro.