Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Progettazione e implementazione di un'app Android per suggerire all'utente ricette volte alla lotta allo spreco alimentare

Design and implementation of an Android app for providing users with recipes conceived to reduce food waste

Relatore

Prof. Domenico Ursino

Candidato

Francesco Alborino

Anno accademico 2018-2019

A coloro che vivono con la consapevolezza di ciò che sono stati e con la determinazione di ciò che vorranno diventare.

Indice

Elenco delle figure

Elenco dei listati

Introduzione

Tra i problemi più sottovalutati, ma più preoccupanti, dell'attuale scenario si classifica lo spreco alimentare. Nella vita quotidiana, non siamo abituati a riflettere sulle conseguenze delle nostre azioni; siamo portati a pensare che se, per una volta, gettiamo un alimento prossimo alla scadenza non succede niente, ma non ci rendiamo conto che, se ognuno buttasse nel cestino un alimento, si sprecherebbe una quantità di cibo impressionante. Tutto ciò senza neanche pensare alla quantità di alimenti gettati dai supermercati e negozi di alimentari che si ritrovano a buttare cibo ormai scaduto, rimasto invenduto. È necessario fare qualcosa nel nostro piccolo, nella nostra quotidianità, per cercare di invertire questa tendenza allo spreco, partendo dal basso e da quello che ognuno di noi può fare, senza troppo sforzo.

EAToday è un'applicazione pensata per operare in questo contesto, cercando di dare un contributo alla lotta allo spreco alimentare, un problema che, negli ultimi anni, sta dilagando e sta diventando sempre più serio. Troppi supermercati e negozi, tra cui anche fruttivendoli, macellerie, panifici, etc. gettano tra i rifuiti alimenti scaduti che nessuno ha comprato. Alcuni di essi, per risolvere questo problema, adottano la politica di proporre al pubblico ad un prezzo scontato i prodotti che si avvicinano alla data di scadenza, sperando di invogliare le persone a comprarli per evitare che vengano gettati; spesso, però, purtroppo, molti alimenti finiscono comunque sprecati, perchè invenduti.

Quest'app nasce proprio con l'intento di cercare di dare un contributo positivo a questa situazione, combattendo lo spreco alimentare e sensibilizzando le persone sull'argomento. È, innanzitutto, necessario far comprendere alle persone che un alimento che si avvicina alla data di scadenza è ancora buono da mangiare e che quindi, se si sta facendo una spesa a breve termine, è bene preferire questi prodotti a quelli con scadenza ancora lontana. Per far ciò, l'idea che si trova alla base del progetto è quella di proporre i prodotti in scadenza di più punti vendita, in modo che gli utenti iscritti possano acquistarli e ricerveli comodamente a casa. Questi prodotti vengono combinati in ricette e suggeriti agli utenti, basandosi sulla distanza dello stesso dai punti vendita e filtrando le ricette in base a dei parametri indicati dall'utente in persona.

L'obiettivo di EAToday è, quindi, quello di fornire un servizio pensato per coloro che hanno piacere di dedicare del tempo alla preparazione di ricette diverse, permettendo loro di ricevere gli ingredienti comodamente a casa. Ovviamente, gli

alimenti che vengono proposti sono quelli in scadenza nei negozi vicini, proprio per incentivare l'acquisto di prodotti che, altrimenti, verrebbero gettati via. Si può definire quest'app come un punto d'incontro tra la comodità di ricevere la spesa direttamente a casa, e la voglia di fare qualcosa nel nostro quotidiano per sovvertire questa tendenza allo spreco di cibo. Utilizzare quest'app è un piccolo passo che ognuno di noi può fare per contribuire a risolvere questo preoccupante problema; un contributo che non reca grande sforzo, in quanto non richiede nient'altro che ordinare la spesa tramite un'applicazione. Utilizziamo, ormai, la tecnologia ogni giorno e per qualunque cosa; perchè non utilizzarla anche per visualizzare ricette, scegliere alimenti, ordinarli e pagarli online, se tutto ciò può aiutare a specare meno cibo? L'idea consiste, perciò, nello sfruttare la tecnologia di cui tutti noi disponiamo, unendola alle consegne a domicilio, oggigiorno sempre più frequenti e utilizzate. Tutto ciò può rappresentare un ottimo incentivo per le persone a comprare alimenti in scadenza, ma che arriveranno comodamente a casa. A tutto ciò si aggiunge la possibilità di filtrare le ricette per ingredienti, in modo da poter consumare eventuali avanzi o prodotti già presenti in casa, agendo ancora di più contro lo spreco alimentare.

EAToday è, quindi, un'applicazione che ha grande potenziale sotto vari punti di vista: in primis è una grande risorsa dal punto di vista del business, in quanto si sta agendo su un campo ancora poco sondato perchè messo in risalto solo di recente, ma ha anche una grande importanza culturale, ponendosi nell'ottica di una maggiore sensibilizzazione ed educazione della popolazione.

EAToday ha grande importanza simbolica anche per un'altra ragione. Quest'applicazione è la prova che il mondo ha bisogno di giovani con la mente creativa per risolvere i problemi odierni. Servono nuove soluzioni, anche un po' fuori dagli schemi, per dare un contributo positivo a tendenze gravi. Se con questa applicazione si riesce a dare un apporto al problema dello spreco del cibo, probabilmente altre idee creative potranno risolvere anche altri problemi gravi e urgenti.

La tesi è strutturata come di seguito specificato:

- Il primo capitolo esporrà un breve excursus sulla nascita ed evoluzione degli smartphone e dei principali sistemi operativi in essi presenti; inoltre saranno analizzate le tecnologie di navigazione outdoor e indoor, che è possibile implementare su questi dispositivi.
- Il secondo capitolo, dopo una breve descrizione generale dell'app EAToday, analizzerà le ragioni che hanno portato allo sviluppo del progetto, descrivendone i requisiti e le funzionalità da fornire agli utenti.
- Il terzo capitolo illustrerà la progettazione dell'app EAToday, sviluppando lo schema E/R del database e i wireframe dell'app. Tale attività consentirà di avere un punto di riferimento costante durante la successiva fase di implementazione.
- Nel quarto capitolo verrà trattata la fase di implementazione per realizzare concretamente l'applicazione; in particolare, verrà illustrato tramite lo sviluppo del back-end su Altervista e del front-end su Android Studio. Successivamente, attraverso degli screenshot, sarà stilato un piccolo manuale utente che illustrerà le principali funzionalità dell'applicazione.
- Il quinto capitolo trarrà alcune conclusioni in merito al lavoro svolto, analizzerà i punti di forza e di debolezza, e darà uno sguardo alle implementazioni future.

1

Caratteristiche generali di Android

Android ormai è il sistema operativo più diffuso creato per dispositivi mobili. La base del suo successo si potrebbe trovare nella sua fluidità e velocità, che hanno conquistato il cuore di milioni di persone, per il suo design unico e per l'androide stilizzato, ovvero il suo logo, il quale rappresenta la firma del prodotto. Oggi un telefono non è niente senza Android.

1.1 L'impatto di Android

Aziende di livello mondiale come *Samsung, Xiaomi, LG, Huawei, HTC* e altri player del settore utilizzano da diversi anni il sistema operativo Android nei loro dispositivi mobili, portando tale sistema operativo ad avere un grande impatto mondiale negli ultimi anni.

1.1.1 Gli inizi

Nell'ottobre 2003 viene fondata la *Android Inc.* per lo sviluppo di ciò che erano stati definiti: "... dispositivi cellulari più consapevoli della posizione e delle preferenze del loro proprietario". Quindi si potrebbe affermare che Android sia nato in quel periodo, ma, in realtà, qui è nato solo il concetto descritto nella citazione precedente. Infatti Android, per come è conosciuto oggi, nasce il 17 agosto 2005, quando *Google* acquistò l'*Android Inc.*, per sviluppare un sistema operativo per dispositivi mobili basato sul kernel di Linux. Android è il risultato di milioni investiti da *Google*.

La presentazione ufficiale del "robottino verde" avviene il 5 novembre 2007 dalla neonata OHA (Open Handset Alliance), un consorzio di aziende del settore Hi Tech, tra le quali *HTC, Samsung e Google*, ovviamente. Il primo dispositivo equipaggiato con Android lanciato sul mercato è l'*HTC Dream*, il 22 ottobre 2008. In seguito si susseguiranno una serie di aggiornamenti per migliorarne le prestazioni e per eliminare eventuali problemi di sicurezza rispetto alle precedenti versioni. Dalla Versione 1.5, ogni aggiornamento prenderà il nome di un dolce e sarà caratterizzato da un proprio particolare logo (Figura 1.1), fino ad arrivare all'attuale aggiornamento "Pie", corrispondente alla Versione 9.0.

Figura 1.1: Loghi delle versioni di Android in ordine di uscita

1.1.2 La grande espansione

Prima di Android, gli smartphone erano telefoni che potevano inviare e-mail e consentire all'utente di accedere al Web. Erano prodotti principalmente da *BlackBerry* e da *Apple*, e limitavano la loro piattaforma solo ai propri dispositivi, tra i quali il più comune era il Blackberry Bold, che governava il mercato in quel momento. Ma Android ha consentito alle piccole aziende, tra cui Micromax, di lanciare i loro smartphone sul mercato ad un prezzo molto basso, permettendo a milioni di persone di acquistarne uno, mentre fino a quel momento gli smartphone erano appannaggio solo delle persone benestanti. Nel tempo il termine "smartphone" è cambiato: ora, ogni volta che qualcuno sente questo termine, visualizza l'immagine di un telefono touchscreen.

Grazie alla sua popolarità, il numero di sviluppatori che lavorano a questa piattaforma è più elevato rispetto a tutte le altre. Questo è il motivo per cui la maggior parte delle app popolari, in un primo momento, sono state sviluppate solo per Android, e le versioni per le altre piattaforme sono state lanciate solo successivamente. Prima di Android, non tutti potevano diventare sviluppatori di app, ma ora basta iscriversi a Google Play per $ 25 e pubblicare delle app.

Le cause della grande espansione di Android sono molteplici, tra queste le principali che lo hanno portato al successo sono le seguenti:

- *Material Design*: Lo stile grafico adottato da Android è definito Material Design, uno stile apprezzato e unico grazie al suo minimalismo. Tra gli elementi più interessanti c'è il menù a tendina che, una volta attivato trascinando il dito dall'estremità superiore a quella inferiore dello schermo, mostrerà tutte le notifiche ricevute. Ripetendo nuovamente lo swipe, si passerà invece al pannello dei toggle rapidi, ovvero il pannello dedicato ad una serie di interruttori che permetteranno di abilitare o disabilitare velocemente alcune funzioni dello smartphone.
- *Giochi e app illimitati*: Prima di Android non era facile trovare giochi e applicazioni. Ora Android presenta un app store, noto come Google Play Store, che fornisce interfacce molto flessibili studiate per essere semplici e comprensibili da chiunque. App come Skype, Facebook, WhatsApp, Telegram, Goggle Drive etc. hanno portato una grande rivoluzione sotto molti aspetti. Tramite Google Play Store è possibile trovare tutte queste app a portata di un semplice click, divise per categorie, tra le quali spicca quella dei giochi per dispositivi mobili.

Quest'ultima tipologia, negli ultimi anni, ha aquisito una fetta di mercato importante, soprattutto dal punto di vista dei ricavi derivanti. Infatti Google Play Store ha milioni di giochi, alcuni a pagamento e altri gratuiti; essi costituiscono una fonte di intrattenimento illimitata, rendendo lo smartphone uno strumento ancora più multi-tasking. Inoltre è possibile trovare online diverse app con estensione .apk, utili per poter usufruire dell'app senza dover necessariamente passare dal Google Play Store.

- *Economico ed efficace*: L'utilizzo di Android da parte di molte aziende produttrici di smartphone ha portato molta concorrenza nel settore con il conseguente abbassamento significativo del prezzo dei dispositivi. Quest'ultimi si diversificano tra loro in molti aspetti, qualcuno punta più sull'autonomia della batteria, qualcun altro sulla fotocamera, altri sulle prestazioni, ma tutti hanno un sistema operativo ottimo e pronto per l'utente, senza che le corrispettive aziende debbano spendere in ricerca e sviluppo per tale componente. Tutto ciò inoltre, ha contribuito allo sviluppo di alcuni paesi, come per esempio l'India.
- *Supporto agli sviluppatori*: Android è una piattaforma grande e affidabile per gli sviluppatori, che creano giochi, app, etc. e, oramai, risulta essere un'ottima opportunità di reddito. Per questo motivo oggi esistono anche, molti corsi di formazione, online e non, sullo sviluppo di Android per insegnare gli elementi essenziali per lo sviluppo di app. Aziende di sviluppo come *Gameloft e EA Sports* sono diventate ottimi player del settore anche grazie alla piattaforma che *Google* è riuscita a fornire loro. I loro principali ricavi derivano principalmente dagli utenti stessi, ovvero dall'acquisto dell'app o dagli acquisti effettuati tramite la stessa app, oppure anche da terzi, ovvero con la pubblicità che alcune società permettono di inserire all'interno dell'app (come, ad esempio, *Google* stessa, con il servizio di Google AdSense).

1.1.3 Le conseguenze in tutto il mondo

Android è stato originariamente creato per offrire servizi di ogni tipo agli utenti, in modo che essi possano migliorare la qualità della loro vita, per esempio risparmiando tempo. Tuttora l'obiettivo è stato ampiamente raggiunto, e si può affermare che molte azioni quotidiane si siano semplificate grazie ai dispositivi Android.

Android ha attualmente più di due miliardi di utenti attivi, il che significa che, con il suo basso costo, ha permesso a molte persone di possedere uno smartphone, cosa impensabile fino a poco più di dieci anni fa. In principio, è stato progettato per dispositivi come telefoni touchscreen e tablet, ma ora è andato ben oltre! Attualmente Android è presente come sistema operativo per televisori, console di videogiochi, orologi digitali e molti altri dispositivi elettronici.

Le tante diverse applicazioni (note come app) sono progettate continuamente per soddisfare le esigenze delle persone. Alcuni dei servizi che oramai vengono garantiti dalle app includono i servizi bancari online, gli aggiornamenti di notizie, gli avvisi meteorologici e i giochi, tra cui quelli di apprendimento per i bambini. Molti servizi di ristorazione hanno app tramite le quali ordinare cibo direttamente dal telefono, per poi pagare e ricevere la consegna a casa. Per non parlare delle chat, delle videochat etc. che permettono a persone distanti pochi kilometri o qualche ora di fuso orario, di sentirsi e restare in contatto. A proposito di kilometri, mentre prima

le indicazioni stradali potevano essere complicate, ora l'utilizzo di uno smartphone con GPS e Google Maps permette a chiunque di spostarsi in modo semplice. Queste app guidano gli utenti passo passo, diventando a tutti gli effetti un navigatore.

Android, con la sua duttilità, è riuscito a conquistare anche il mercato degli orologi digitali, aggiungendo svariati benefici. I più interessanti riguardano il controllo della salute fisica, tramite la misurazione della frequenza cardiaca, il conteggio dei passi e il controllo della qualità del sonno.

1.2 L'architettura di Android

Il sistema operativo Android risulta semplice e brillante, ma anche elegante: sono presenti ottime scorciatoie molto fluide e utili. L'efficace usabilità di Android è data da tre elementi principali: la *lockscreen*, la *homescreen* e l'*app drawer*.

La *lockscreen* è la schermata di blocco del dispositivo. Il suo scopo è quello di tenere, appunto, bloccato il dispositivo nell'attesa di ricevere istruzioni, in modo da evitare l'avvio accidentale di applicazioni, chiamate o quant'altro. Per accedere alle funzionalità del dispositivo, dunque, è necessario sbloccarne la *lockscreen*.

Da qui si accede alla *homescreen* che può essere pensata come il desktop dello smartphone. Questa, infatti, altro non è che uno spazio molto ampio, costituito da un numero di pagine personalizzabile e sfogliabile grazie al tocco delle dita.

Infine, l'*app drawer* raggruppa tutte le applicazioni installate sul dispositivo; al suo interno si trova, anche, l'area impostazioni del dispositivo grazie alla quale poter configurare il proprio smartphone.

Tutto ciò è merito dell'ottima architettura che si cela dietro a questi tre elementi, realizzata con una serie di componenti divisi per strati, in cui i livelli inferiori fungono da base per i livelli superiori, offrendo un più alto grado di astrazione.

1.2.1 Componenti principali

Android comprende tutto lo stack degli strumenti per la creazione di applicazioni (Figura 1.2), tra cui un sistema operativo, un insieme di librerie native per le funzionalità core della piattaforma, un'implementazione della *Virtual Machine* e un insieme di librerie *Java*.

Il kernel di Linux

Il layer di più basso livello è rappresentato dal kernel Linux. Risulta molto utile la presenza di driver per la gestione delle periferiche multimediali, del display, della connessione Wi-Fi e dell'alimentazione. Molto importante è, anche, la presenza di un driver dedicato alla gestione della comunicazione tra processi diversi (IPC).

Librerie native

Sopra al kernel di Linux si trova un livello che contiene un insieme di librerie native realizzate in C e C++, che rappresentano il core vero e proprio di Android. Tra le più importanti citiamo:

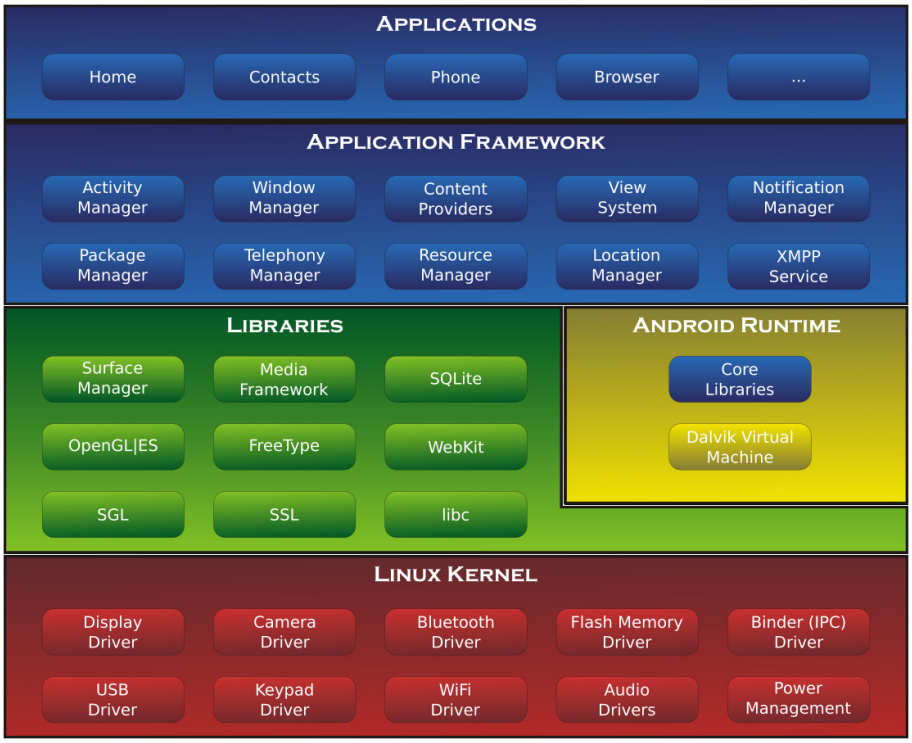


Figura 1.2: Struttura a strati di Android

- *Surface Manager*: componente fondamentale in quanto ha la responsabilità di gestire le view, ovvero ciò da cui un'interfaccia grafica è composta.
- *SGL*: una libreria in C++ che insieme alle OpenGL, costituisce il motore grafico di Android. Mentre per la grafica 3D ci si appoggia all'*Open GL*, per quella 2D viene utilizzato un motore ottimizzato chiamato, appunto, *SGL*.
- *Media Framework*: gestisce la multimedialità. È in grado di gestire i diversi *CODEC* per i vari formati di acquisizione e di riprodurre audio e video.
- *SQLite*: sistema che permette di memorizzare in modo persistente sul dispositivo un certo insieme di informazioni. È una libreria "in-process" che implementa un DBMS relazionale caratterizzato dal fatto di essere molto compatto, diretto, di non necessitare alcuna configurazione, e soprattutto, transazionale.
- *WebKit*: browser engine open source basato sulle tecnologie *HTML, CSS, JavaScript e DOM* (essendo un browser engine andrà integrato in diversi tipi di applicazioni).
- *SSL*: libreria per la gestione dei *Secure Socket Layer*, grazie alla quale Android risulta avere un'ottima sicurezza.

Application Framework

Tutte le librerie viste finora vengono poi utilizzate da un insieme di componenti di più alto livello che costituiscono l'Application Framework. Tutte le applicazioni

per Android adottano lo stesso Application Framework e, come tali, possono essere estese, modificate o sostituite. Il tutto non è altro che un insieme di API e componenti per l'esecuzione di funzionalità ben precise e di fondamentale importanza in ciascuna applicazione Android. Tra le più importanti, citiamo:

- *Activity Manager*: non solo permette la visualizzazione o la raccolta di informazioni ma, in modo più generico, è lo strumento fondamentale attraverso il quale l'utente interagisce con l'applicazione.
- *Package Manager*: gestisce i processi di installazione delle applicazioni nei dispositivi.
- *Window Manager*: permette di gestire le finestre delle diverse applicazioni, sullo schermo del dispositivo, anche se queste vengono gestite da processi differenti.
- *Content Provider*: gestisce la condivisione di informazioni tra i vari processi.
- *View System*: gestisce la renderizzazione dei componenti e tutti gli eventi da essi associati.

1.2.2 Software per lo sviluppo

Negli ultimi anni sono stati sviluppati diversi approcci che permettono di sviluppare app per smartphone. Questi consentono di realizzare tre tipi di app: native, web app e ibride.

Le prime rappresentano il modo più tradizionale per implementare un'app, utilizzando i vari SDK e framework offerti dai produttori stessi del sistema operativo. Un'app nativa, generalmente, ha il vantaggio di poter accedere direttamente alle funzionalità del sistema operativo, senza aver bisogno di astrazione o middleware aggiuntivi. Per questo motivo, questa soluzione è preferita da molti sviluppatori che abbiano a cuore le performance dell'app. I più importanti ambienti di sviluppo per il mondo Android sono *Eclipse IDE e Android Studio*. *Eclipse IDE* è un framework per lo sviluppo e la compilazione di applicazioni basate sul linguaggio *Java*. Se utilizzato in combinazione con un componente aggiuntivo gratuito denominato *Android Development Tools (ADT)*, consente di sviluppare app Android e di testarle in maniera abbastanza celere. Non è semplicissimo da usare a primo acchito, ma basta fare un po' di pratica per acquisire una certa dimestichezza. È disponibile per Windows, macOS e Linux. *Android Studio* verrà analizzato dettagliatamente in seguito.

Le web application, invece, non sono altro che siti web responsive e particolarmente ottimizzati per il mobile, che, però, hanno poco a che vedere con le app. Tali web application richiedono, infatti, un qualsiasi browser mobile; tuttavia le funzionalità utilizzabili sono limitate, e le prestazioni che possono essere garantite sono, in generale, inferiori.

Infine, le app ibride consistono nell'uso di un componente nativo in grado di visualizzare pagine web e di una web application che viene automaticamente visualizzata tramite tale componente nativo. Negli ultimi anni si sono affermati diversi framework per lo sviluppo di app multipiattaforma che creano app ibride. Esse utilizzano un unico linguaggio di programmazione per la definizione dell'app, che può, poi, essere convertita in una implementazione adatta alle varie piattaforme supportate (tra cui sono sempre presenti Android ed iOS). Ne è un esempio il C# che è un linguaggio ad oggetti più ad alto livello rispetto a C/C++: usato con *Xamarin*

all'interno di *Visual Studio* è un'ottima soluzione per sviluppare app cross platform. Un altro esempio è *Corona*, che rappresenta un'altra opzione valida per sviluppare app Android. Utilizzando il linguaggio *LUA*, più semplice di *Java*, e sfruttando il *Corona SDK* (*Software Development Kit*), è particolarmente usato nella creazione di giochi, ma può essere utilizzato anche per sviluppare altri tipi di applicazioni.

Focus su Android Studio

Figura 1.3: Logo di Android Studio

Android Studio (il cui logo è riportato in Figura 1.3) è un software messo a disposizione direttamente da *Google* e consente agli sviluppatori di realizzare app destinate al sistema operativo Android, tramite un apposito ambiente di sviluppo integrato (IDE). Basato sul software di *JetBrains IntelliJ IDEA*, *Android Studio* sostituisce gli *Android Development Tools (ADT) di Eclipse*, diventando l'IDE primario di *Google* per lo sviluppo nativo di applicazioni Android.

È stato annunciato il 16 maggio 2013 ed è disponibile gratuitamente per Windows, Mac OS X e Linux, sotto licenza *Apache 2.0*.

Uno dei suoi obiettivi principali è quello di rendere più produttivo il lavoro del programmatore offrendo un editor snello, con template di applicazioni già pronti e molti tool di supporto, come il meccanismo di preview dei layout: l'aspetto dell'interfaccia viene mostrato in anteprima, adattato realisticamente ad uno specifico modello di dispositivo. *Android Studio* include, inoltre, *Gradle*, un ottimo strumento per la build automation, molto flessibile ed erede di tutte le principali caratteristiche dei suoi predecessori, come *Apache Ant e Maven*.

2

Analisi dei requisiti

Il seguente capitolo, dopo una breve descrizione generale dell'app EAToday, analizza, in sintesi, le ragioni che hanno portato allo sviluppo del progetto, descrivendone i requisiti e le funzionalità da fornire agli utenti.

2.1 Descrizione generale di EAToday

EAToday è un'applicazione pensata per contrastare lo spreco alimentare, un problema che, negli ultimi anni, sta dilagando e sta diventando sempre più serio. In vari supermercati, nei negozi, tra cui anche fruttivendoli, macellerie, panifici, etc. succede spesso che molti prodotti vengano gettati via perchè scaduti e rimasti invenduti. Alcuni supermercati, per risolvere questo problema, adottano la politica di proporre al pubblico ad un prezzo scontato i prodotti che si avvicinano alla data di scadenza, sperando di invogliare le persone a comprarli per evitare che vengano gettati; spesso, però, purtroppo, molti alimenti finiscono comunque sprecati, perchè invenduti.

Quest'app nasce proprio con l'intento di cercare di dare un contributo positivo a questa situazione, combattendo lo spreco alimentare e sensibilizzando le persone sull'argomento. Per far ciò, l'idea che si trova alla base del progetto è quella di combinare i prodotti in scadenza di più punti vendita, in modo che gli utenti iscritti possano acquistarli e ricerveli comodamente a casa. Questi prodotti vengono proposti nell'applicazione attraverso delle ricette, selezionate in modo mirato in base ai prodotti disponibili. Il tutto è contornato da diverse funzionalità, utili per rendere quanto migliore possibile l'esperienza utente. Un possibile Flyer, molto sintetico, ma esplicativo, è rappresentanto in Figura 2.1.

2.1.1 Indagine di mercato

Essendo un'idea di business innovativa per una possibile Start Up, questo progetto necessita di un'indagine di mercato, utile per individuare il mercato di riferimento, analizzarlo, e stabilire il target interessato.

Le indagini di mercato sono fondamentali per condurre una ricerca di marketing; infatti, queste misurano i sentimenti e le preferenze dei clienti in un dato mercato.

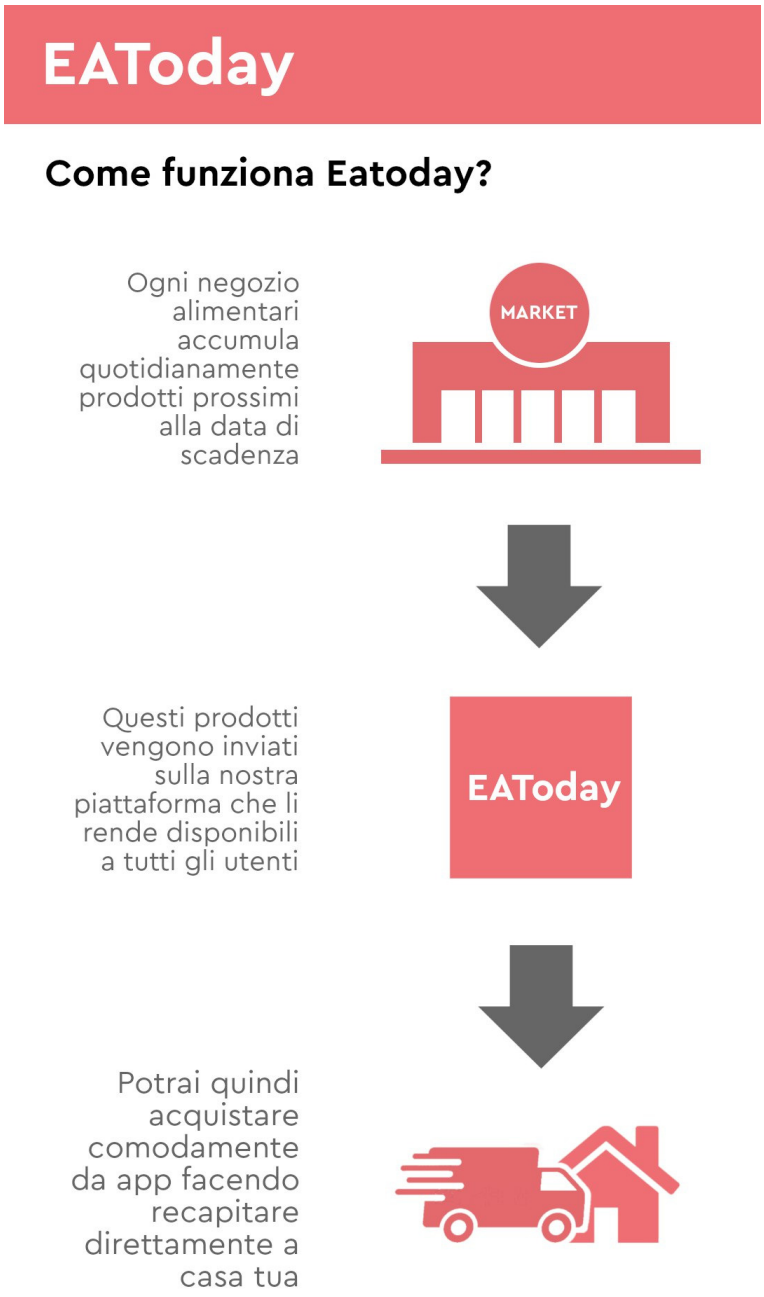


Figura 2.1: Possibile flyer di EAToday

Nel nostro caso, utilizzeremo un campione molto piccolo (circa 200 persone), ma

i risultati saranno comunque utili per avere un primo riscontro sulle decisioni da prendere per poter avviare il progetto, ma, soprattutto, per tramutare le necessità dei consumatori in funzionalità dell'app, ovvero per stabilire i requisiti fondamentali dell'app da realizzare.

Portata e dimensioni del mercato

Per comprendere meglio l'entità di questo problema, basta consultare un po' di dati relativi agli ultimi anni. Il mercato alimentare in Italia genera annualmente più di 10 milioni di tonnellate di cibo che vengono gettate via, l'equivalente di 20 tonnellate per minuto, ovvero 317 kg ogni secondo. Questo spreco aumenta ogni giorno, e in termini di spesa corrisponde a 17 miliardi di euro annui. Sono circa 700 euro l'anno i soldi spesi da ogni famiglia per acquistare del cibo che poi, però, non verrà consumato. Negli ultimi anni, forse per educazione o forse per crisi, la situazione in Italia è un po' migliorata, passando da una persona su due che dichiarava di gettare cibo tra i rfiuti, all'attuale uno su cento. Mentre in Italia si può notare un miglioramento, anche se non sufficiente, bisogna comunque tenere presente i dati globali, molto drammatici. Dal 1974 lo spreco alimentare è aumentato del 50%, ma solo di recente si è iniziato a parlare di questo problema. Attualmente, un terzo del cibo prodotto non viene consumato ma gettato.

Introdursi in questa situazione, cercando di alleviarla, o per lo meno, cercando di educare la popolazione in modo tale da renderla più consapevole del problema economico, oltre che ambientale, potrebbe essere considerata un'ottima opporturnità da cogliere per far partire questo progetto.

Gli articoli di testate giornalistiche e gli interventi delle persone del settore, effettuati negli ultimi anni, hanno evidenziato il problema, portando molte persone a ragionare sull'argomento. Tutto ciò ha aumentato la portata del target da raggiungere per diffondere il progetto.

Sondaggio

Utilizzando i moduli forniti da *Google* è stato possibile realizzare un sondaggio, per riuscire a ottenere i primi dati su cui basarsi per scegliere le funzionalità da inserire nell'applicaizione.

Un questionario è stato somministrato a circa duecento persone tramite una opportuna form online. Le domande erano strutturate in modo da inquadrare in un primo momento il tipo di persona intervistata (chiedendo dati personali) e, in un secondo momento, ponendo domande relative al suo rapporto con la spesa e alla preparazione degli alimenti.

Segmento di clientela

Per riuscire a definire il segmento di clientela, sono state fondamentali due domande presenti all'interno del questionario, poi combinate con le domande che definivano personalmente ogni soggetto.

Il risultato è stato che, preso un campione di individui tra i 18 e i 60 anni, con una maggioranza di giovani, più del *50%* delle persone utilizzano app, blog o

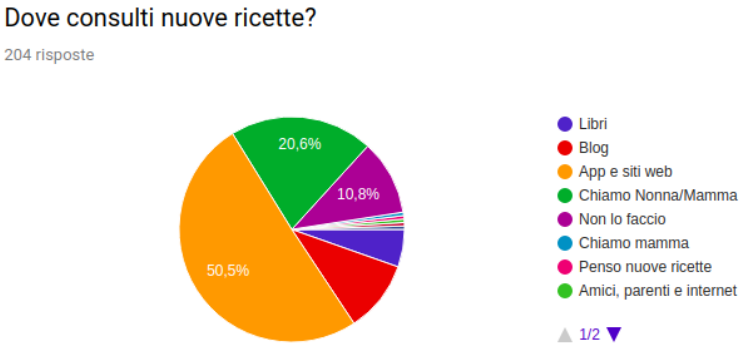


Figura 2.2: Domanda relativa alla consultazione delle ricette

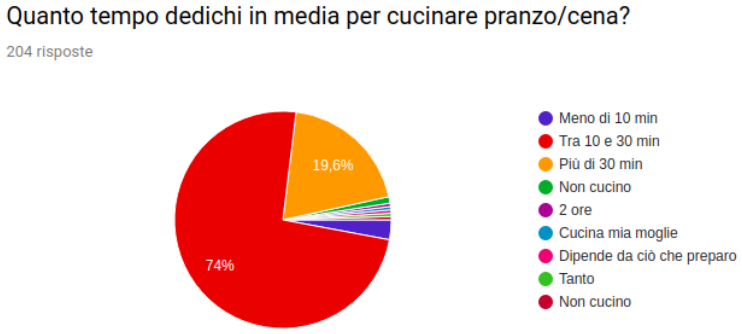


Figura 2.3: Domanda relativa al tempo di preparazione

Internet in generale per ricercare ricette e dilettarsi nello sperimentare nuovi tipi di pietanze (Figura 2.2).

Inoltre, più del *75%* impiega dai 10 ai 30 minuti per cucinare, accanto a questa consistente maggioranza, è prensente una significativa parte delle persone, quasi il *20%*, disposta a spendere più di 30 minuti per preparare il pranzo o la cena (Figura 2.3).

In conclusione, un esempio di target perfetto è rappresentato da una persona giovane che per gran parte del giorno si trova fuori casa, senza riuscire a fare spesso la spesa, ma che comunque avrebbe voglia, soprattutto la sera, di dedicare del tempo a cucinare, spendendo poco e sperimentando nuove ricette, utilizzando un'applicazione come guida.

2.1.2 Proposta di valore

L'obiettivo di EAToday è fornire un servizio pensato per a coloro che hanno piacere di dedicare del tempo alla preparazione di ricette diverse, permettendo loro di ricevere gli ingredienti comodamente a casa. Gli alimenti che vengono consigliati sono, ovviamente, quelli in scadenza nei negozi vicini, proprio per incentivare l'acquisto di prodotti che, altrimenti, verrebbero gettati via. Si può dire che l'app si pone

come punto d'incontro tra il piacere della buona cucina e la voglia di fare qualcosa, nel piccolo del quotidiano, per contribuire alla risoluzione di questo problema dello spreco del cibo. A questo si aggiunge la possibilità di filtrare le ricette per ingredienti, in modo da poter consumare eventuali avanzi o prodotti già presenti in casa, agendo ancora di più contro lo spreco alimentare.

In poche parole, EAToday non ha del potenziale solo dal punto di vista del business, in quanto si sta agendo su un campo ancora poco sondato perchè messo in risalto solo di recente, ma ha anche una grande importanza culturale, ponendosi nell'ottica di una maggiore sensibilizzazione ed educazione della popolazione.

2.2 Funzionalità dell'app

Seguirà, ora, un'analisi dei requisiti funzionali e non funzionali relativi all'applicazione. I requisiti funzionali descrivono i servizi, o funzioni, offerti dall'applicazione, mentre i requisiti non funzionali rappresentano vincoli sui servizi offerti dall'applicazione stessa.

2.2.1 Requisiti funzionali

"EAToday" garantirà ai suoi utilizzatori i seguenti servizi:

1. visualizzare le ricette e ricercarle per nome;
2. filtrare le ricette in base al tipo di pasto, alla difficoltà e agli ingredienti desiderati e indesiderati dall'utente;
3. permettere all'utente, se registrato, di ordinare gli ingredienti per una ricetta;
4. offrire la possibilità di registrazione, con tanto di modifica del profilo, ai servizi offerti dell'app;
5. permettere l'autenticazione all'interno dell'app.

2.2.2 Requisiti non funzionali

I requisiti non funzionali saranno i seguenti:

1. l'applicazione dovrà essere realizzata mediante Android Studio.
2. i linguaggi di programmazione utilizzati per la realizzazione dell'applicazione dovranno essere Java e XML.

2.3 Diagramma dei casi d'uso

Il diagramma dei casi d'uso relativo all'app EAToday viene mostrato nella Figura 2.4

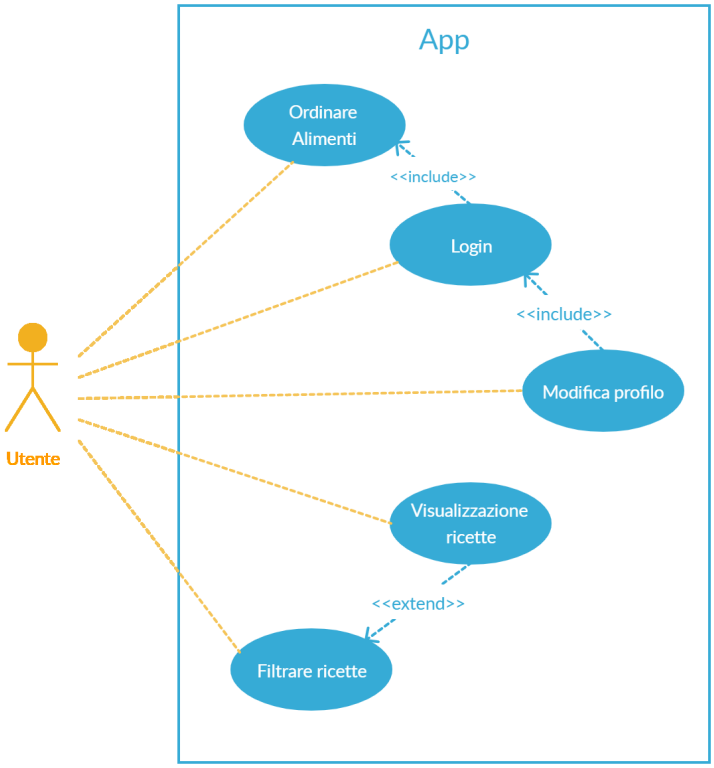


Figura 2.4: Diagramma dei casi d'uso dell'app EAToday

Dall'analisi di questa figura si può notare che i casi d'uso relativi all'applicazione sono cinque:

- *Login*: il login permette all'utente di accedere ai servizi dell'applicazione riservati agli utenti registrati. Come si nota in Figura 2.4, tali servizi sono due: *Ordinare alimenti* e *Modifica profilo*. Nel caso in cui il cliente non si sia mai registrato al servizio, dovrà effettuare la registrazione, in cui inserire i propri dati.
- *Modifica profilo*: permette all'utente di modificare i propri dati, inseriti al momento della registrazione. Tale caso d'uso richiede che l'utente si sia autenticato, ed è utile per effettuare una qualsiasi modifica dei dati personali avvenuta successivamente alla registrazione.
- *Visualizzazione ricette*: essa non è altro che la *home page* dell'applicazione, dove verranno visualizzare le varie ricette presenti nel database con alimenti disponibili nei negozi vicini. Non è richiesto il login per questo caso d'uso.

- *Filtrare ricette*: caso d'uso esteso dal precedente; permette di filtrare le ricette visibili nella home page, seguendo una form presente in un'altra pagina.
- *Ordinare alimenti*: l'utente, una volta scelta la ricetta desiderata, potrà ordinare tutti, o una parte, degli ingredienti presenti. Se autenticato, egli potrà ordinare gli ingredienti, ricevendo un messaggio di conferma nel caso sia andato tutto correttemente.

3

Progettazione

In questo capitolo illustreremo la progettazione dell'app oggetto della presente tesi. Tale attività ci consentirà di avere un punto di riferimento costante durante la successiva fase di implementazione.

3.1 Progettazione del database

L'intero progetto necessita di un database online al quale potranno accedere i diversi utenti sia per registrarsi e autenticarsi, sia per avere informazioni quali ricette e ingredienti disponibili, aggiornati sempre in tempo reale. Per questo motivo, il back-end dovrà essere realizzato su un server disponibile online e dovrà contenere informazioni riguardo gli utenti, le ricette, gli ingredienti e i negozi nel quale trovare questi ultimi.

Inoltre, i negozi dovranno inviare al database gli ingredienti disponibili, così che questi possano essere combinati in ricette e resi disponibili agli utenti.

3.1.1 Progettazione concettuale

La progettazione concettuale serve per rappresentare i dati della realtà di interesse in un modello concettuale, sotto forma di concetti e di relazioni tra essi. Durante questa fase vengono individuate le entità della realtà di riferimento e le relazioni che esistono tra di esse. Quindi, un'entità non è altro che un concetto con un'esistenza autonoma, mentre un'istanza di un'entità è un elemento reale che si può rappresentare all'interno dell'entità stessa.

Il diagramma E/R permette di rappresentare il contenuto informativo di un database. Per completezza, tutti gli elementi rappresentati nel diagramma verranno, successivamente, descritti in maniera più approfondita tramite il Dizionario delle Entità e delle Relazioni. In Figura 3.1 viene mostrato il diagramma E/R della realtà di interesse.

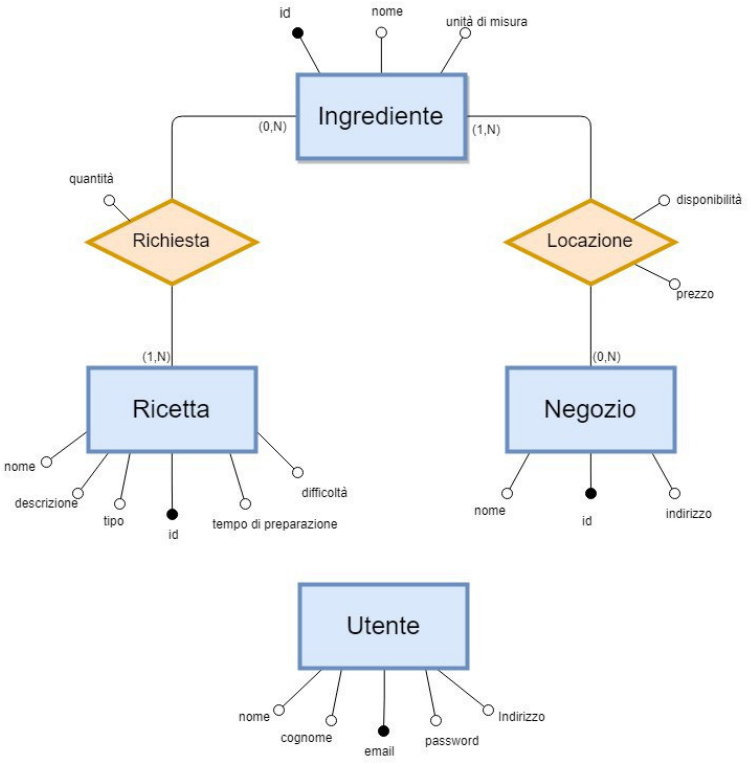


Figura 3.1: Schema E/R della realtà di interesse

Descrizione dello schema E/R

Lo schema E/R (Figura 3.1) è diviso in entità e relazioni rappresentanti le informazioni utili per poter mostrare le ricette e gli ingredienti da acquistare agli utenti, mentre l'entità Utente è utile per conservare in maniera persistente i dati dell'utente stesso e per permettere il login.

Queste entità e queste relazioni esprimono proprietà elementari associate ad una entità o ad una associazione.

In breve, i principali costrutti e relazioni di uno schema E/R sono:

- *Entità*: rappresenta una classe di oggetti con proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse. In uno schema E/R, ogni entità viene identificata univocamente da un nome al singolare e viene rappresentata graficamente tramite un rettangolo con il nome dell'entità al suo interno.
- *Relationship*: rappresenta un legame logico tra due o più entità. Il numero di entità legate definisce il grado dell'associazione: un buono schema E/R è caratterizzato da una prevalenza di relazioni binarie, ossia di grado due. Per il nome dell'associazione si opta generalmente per un sostantivo singolare, preferibilmente un nome espressivo, evitando di dare un verso alla stessa. Di norma una relazione viene rappresentata graficamente da un rombo contenente il suo nome.

- *Attributo*: permette di descrivere le entità e le corrispettive relationship; la scelta degli attributi riflette il livello di dettaglio con il quale si vogliono rappresentare le informazioni. Per ogni entità va scelto un particolare attributo, chiamato identificatore, che può essere interno o esterno, utile per identificare univocamente ogni occorrenza dell'entità.

Dizionario delle Entità e delle Relazioni

La progettazione concettuale comprende, come detto precedentemente, la presenza di un dizionario dei dati, costruito sia per le entità che per le relationship, il cui scopo è di descrivere e di spiegare in maniera chiara ed esaustiva ogni elemento del diagramma E/R realizzato. In particolare, si specificano, per ogni elemento, la sua descrizione, i suoi attributi e il suo identificatore. Nella Tabelle 3.1 e 3.2, vengono rappresentati, rispettivamente, il Dizionario delle Entità e il Dizionario delle Relazioni del database dell'app EAToday.

Nome entità	*Descrizione*	*Attributi*	*Identificatore*
Utente	Individuo che utilizza l'applicazione dopo aver effettuato la registrazione.	nome (stringa), cognome (stringa), email (stringa), password (stringa), indirizzo (stringa)	email
Ricetta	Indicazione degli ingredienti, delle dosi e delle modalità di confezione con cui preparare pietanze, dolci, conserve, bibite e bevande varie, o anche prodotti non alimentari.	id (numerico), descrizione (stringa), nome (stringa), tipo (stringa), tempo di preparazione (stringa), difficoltà (stringa)	id
Ingrediente	Ogni sostanza che entra nella composizione di una ricetta.	id (numerico), nome (stringa), unità di misura (stringa),	id
Negozio	Locale destinato all'esposizione e alla vendita di merci al pubblico.	id (numerico), nome (stringa), indirizzo (stringa)	id

Tabella 3.1: Dizionario dei dati che descrive le entità dello schema E/R dell'applicazione

3.1.2 Progettazione logica

La progettazione logica permette di effettuare la traduzione dallo schema concettuale in uno schema logico, ed ha lo scopo di rappresentare le infomazioni contenute nello schema concettuale in un formato più vicino alla macchina, garantendo comunque correttezza ed efficienza.

Tutto ciò avverà tramite una ristrutturazione dello schema E/R in un primo momento, e, di seguito, tramite la traduzione di quest'ultimo in uno schema logico.

Nome relationship	*Descrizione*	*Attributi*	*Identificatore*
Richiesta	Indica gli ingredienti che sono richiesti per la preparazione di una determinata ricetta.	idRicetta (numerico), idIngrediente (numerico),quantità (stringa)	idRicetta, idIngrediente
Locazione	Indica la provenienza dei vari ingredienti.	idIngrediente (numerico), idNegozio (numerico), disponibilità (stringa), prezzo (stringa)	idNegozio, idIngrediente

Tabella 3.2: Dizionario dei dati che descrive le relationship dello schema E/R dell'applicazione

Ristrutturazione dello schema E/R

Per effettuare la ristrutturazione dello schema E/R sono necessari i seguenti passaggi:

1. *Analisi delle ridondanze*: si decide se eliminare o meno le ridondanze dello schema in base al costo delle varie operazioni che le coinvolgono, valutando, per ogni ridondanza, se risulta conveniente conservarla rimuoverla.
2. *Eliminazione delle generalizzazioni*: tutte le generalizzazioni dello schema vanno sostituite dai costrutti elementari, in quanto il modello relazionale non permette la loro rappresentazione.
3. *Partizionamento/accorpamento di entità e di relazioni*: si decide se partizionare o accorpare concetti dello schema per garantire efficienza nelle operazioni.
4. *Scelta degli identificatori primari*: tale scelta è essenziale e ci sono delle regole da rispettare; talvolta viene aggiunto un attributo identificatore utilizzando la proprietà `AUTOINCREMENT`, che permette di assegnare un valore crescente univoco per identificare ogni occorrenza di una entità.

Traduzione dello schema E/R

Lo schema concettuale di EAToday risulta molto semplice, almeno per questa prima versione del progetto, per cui non è stato neccessario effettuare alcuna ristrutturazione; per questo motivo, verrà utilizzato lo stesso schema, mostrato in Figura 3.1, per ottenere lo schema relazionale.

Tutte le relazioni, presenti nello schema E/R, sono di tipo "molti a molti" e, per questo motivo, queste hanno come identificatori le chiavi delle entità coinvolte (che formano la chiave primaria).

A questo punto, la traduzione risulta essere:

- **Ricetta**(id, nome, descrizione, tipo, tempo di preparazione, difficoltà)
- **Richiesta**(idRicetta, idIngrediente, quantità)
- **Ingrediente**(id, nome, unità di misura)
- **Locazione**(idIngrediente, idNegozio, disponibilità, prezzo)

- **Negozio**(id, nome, indirizzo)
- **Utente**(email, nome, cognome, password, indirizzo)

3.1.3 Strumenti utilizzati per il back-end

Per gestire la comunicazione tra l'app e un database condiviso da tutti gli utenti (Figura 3.2), è stato necessario utilizzare un servizio che offrisse dello spazio online nel quale organizzare le informazioni all'interno di una base di dati che rispecchiano le esigenze e la struttura appena progettata.

Il servizio scelto è stato *AlterVista.org*.

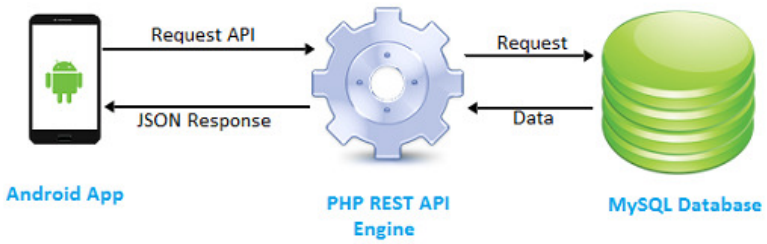


Figura 3.2: Comunicazione tra smartphone Android e database

Altervista.org

Su *AlterVista.org* è possibile creare un sito web con *PHP* e *DBMS MySQL*, tramite *phpMyAdmin*. L'uso dello spazio è libero, ma le risorse a disposizione, come spazio web e traffico mensile, sono limitate inizialmente, ed espandibili tramite l'acquisto o l'inserimento della pubblicità.

La sua principale utilità per l'applicazione EAToday è stata, sostanzialmente, quella di fornire degli *Script PHP* per tradurre le informazioni contenute nel database in una struttura dati *JSON*. Tramite richieste *GET* e *POST* è stato, rispettivamente, possibile ricevere dati dal database, per esempio quelli relativi alle ricette, e inviare dati al database, per esempio quelli relativi a un nuovo utente registrato.

Tutti questi *Script PHP* sono stati divisi in cartelle per questioni organizzative; inoltre, è stata aggiunta una cartella `images`, dove sono state salvate le immagini utili all'applicazione, pronte ad essere caricate dinamicamente direttamente dall'app.

phpMyAdmin

phpMyAdmin è un'applicazione web scritta in *PHP* che consente di amministrare un database *MySQL* tramite un qualsiasi browser.

Esso permette di creare un database "da zero", creare tabelle in un database già esistente ed eseguire operazioni di ottimizzazione su di esse. Tale web-app presenta un feedback sulla creazione delle tabelle per evitare eventuali errori. Sono anche previste delle funzionalità per l'inserimento dei dati (popolazione del database), per le query, per il backup dei dati, etc.

Focus sul JSON

JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma che utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, *Java*, *JavaScript*, *Python*, e molti altri. Tale caratteristica fa di *JSON* un linguaggio ideale per lo scambio di dati.

JSON è basato su due strutture:

- *Un insieme di coppie nome/valore*: in diversi linguaggi, questo è realizzato come un oggetto, un record, una struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.
- *Un elenco ordinato di valori*: nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Queste sono strutture di dati universali. Virtualmente tutti i linguaggi di programmazione moderni li supportano in entrambe le forme.

È sensato che un formato di dati che sia interscambiabile con linguaggi di programmazione debba essere basato su queste strutture. Nel Listato 3.1 si riporta un esempio di file *JSON*; in particolare, viene mostrato un file *JSON* utilizzato nel progetto, ovvero quello che contiene le ricette.

```
{
"id": "3",
"name": "insalata con finocchi",
"type": "vegano",
"difficulty": "facile",
"time": "10",
"description": "Per preparare insalata [...]",
"ingredients": [
  {
    "id_recipe": "3",
    "id_ingredient": "3",
    "quantity": "100 g",
    "name": "insalata",
    "unit": "180 g",
    "availability": "3",
    "price": "1",
    "store": "Simply"
  },
  {
    "id_recipe": "3",
    "id_ingredient": "7",
    "quantity": "50 g",
    "name": "olio extra vergine di oliva",
    "unit": "1 L",
    "availability": "3",
    "price": "2.5",
    "store": "Si con Te"
  }
  ]
}
```

Listato 3.1: Esempio di file JSON per le ricette

3.2 Struttura applicazione

La struttura in Figura 3.3 rappresenta la mappa dei percorsi possibili all'interno dell'applicazione EAToday per raggiungere una determinata schermata.

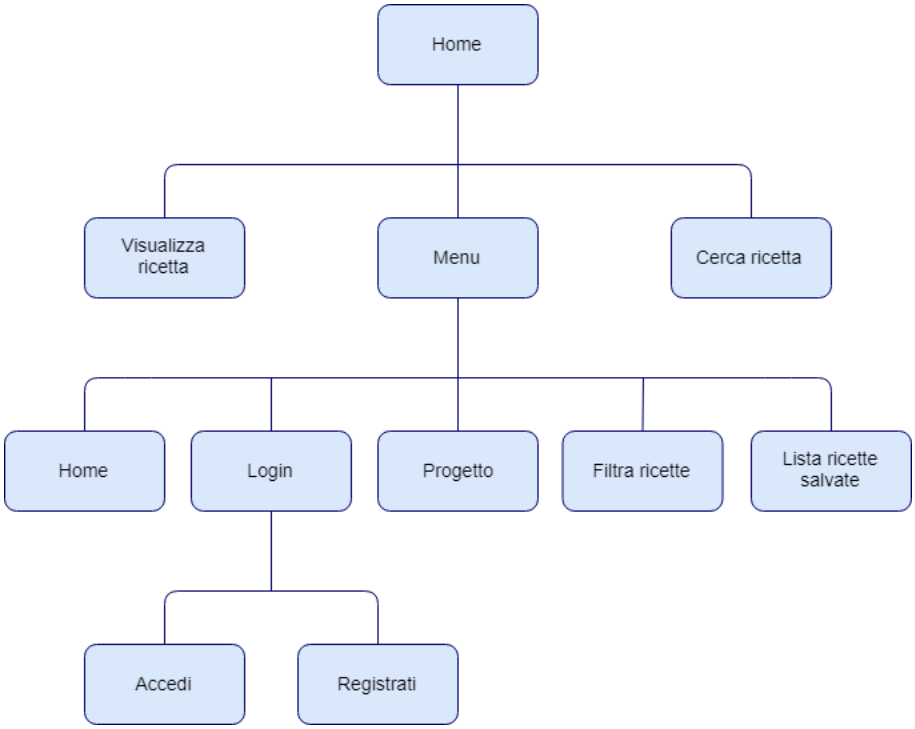


Figura 3.3: Struttura dell'app EAToday

La schermata principale è la home page, nella quale, all'avvio, vengono visualizzate le informazioni principali di ogni ricetta, grazie ad uno *Script PHP* che, tramite una richiesta *GET*, esegue alcune query e scarica tutte le informazioni necessarie dal database.

Come si può notare dalla Figura 3.3, l'utente, a partire dalla home page, può eseguire le seguenti operazioni:

- *Cerca Ricetta*: attraverso la barra di ricerca, l'utente può cercare per nome una ricetta. La schermata iniziale si aggiorna in base alle lettere inserite dall'utente nella barra di ricerca.
- *Visualizza Ricetta*: l'utente, cliccando su una determinata ricetta, potrà visualizzarne gli ingredienti e modalità di preparazione. Si ha, poi, la possibilità di acquistare gli ingredienti di cui si ha bisogno.
- *Menù*: grazie all'apposito pulsante in alto a sinistra, come si nota nella Figura 4.2, un utente non ancora autenticato ha la possibilità di effettuare le seguenti operazioni:

1. *Login*: conduce alla schermata che permette di effettuare il login.
2. *Filtra ricette*: conduce alla schermata dove è possibile definire i criteri per filtrare le varie ricette.
3. *Ricette salvate*: conduce alla schermata per recuperare le ricette salvate.
4. *Progetto*: conduce alla schermata di presentazione dell'intero progetto, utile anche per capire meglio il funzionamento dell'applicazione.

Se l'utente effettua il login, il menù cambia e le operazioni possibili diventano quelle in Figura 3.5; al posto della schermata "Login" si può raggiungere la schermata per la modifica del profilo. Inoltre, nel menù, è presente una voce aggiuntiva per effettuare il "Logout".

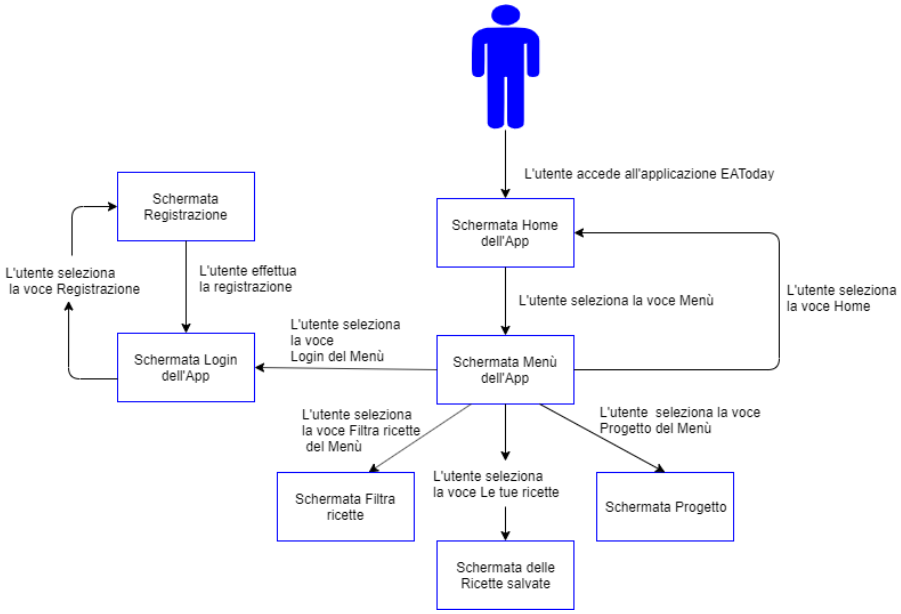


Figura 3.4: Azioni disponibili ad utenti non autenticati

3.2.1 Wireframe

Un wireframe non è altro che una bozza del lavoro che verrà svolto, è un documento a bassa-fedeltà: non sarà navigabile (essendo un'immagine statica) e descriverà la giusta posizione degli elementi all'interno della vista (pagina web).

Può essere considerato come lo scheletro del front-end dell'applicazione web, dove vengono assicurate la giusta posizione dei blocchi principali del contenuto e una corretta struttura delle informazioni e vengono, oltresì, descritte le principali interazioni utente-interfaccia.

Solitamente un wireframe è in bianco e nero, perchè, in questa fase, l'attenzione è focalizzata sugli elementi che caratterizzano il proprio prodotto, come, nel caso

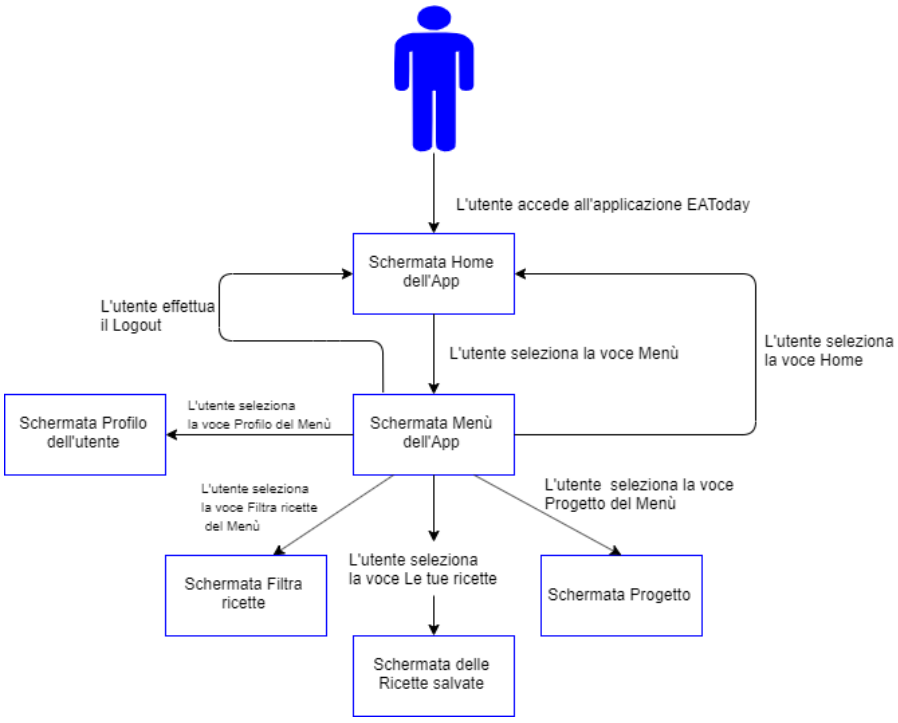


Figura 3.5: Azioni disponibili ad utenti autenticati

di un'app, la posizione dei vari elementi nelle varie schermate. Un wireframe ha la funzione di:

- comunicare l'idea iniziale del progetto.
- focalizzare l'attenzione solo su architettura e contenuti.
- comunicare cosa si vedrà.
- essere la base del prototipo.

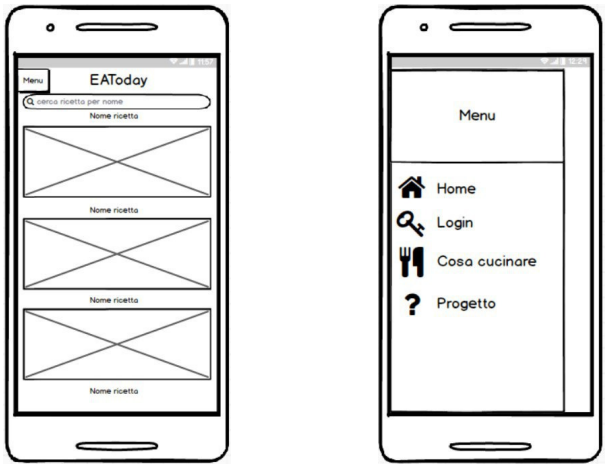
Di seguito vengono rappresentati i wireframe dell'App EAToday:

Schermata Home

Il wireframe in Figura 3.6(a) rappresenta la schermata iniziale dell'applicazione EAToday. In modo semplice vengono rappresentati: le ricette, il pulsante menù, la barra di ricerca e il nome dell'applicazione.

Schermata del Menù

Il wireframe in Figura 3.6(b) rappresenta la schermata con il drawer menù, che si ottiene premendo il pulsante in alto a sinistra. Si può osservare che l'utente avrà a disposizione quattro possibili scelte nel menù.



(a) *"Home"* (b) *"Menù"*

Figura 3.6: Wireframe delle schermate Home e Menù

Schermata Login

Il wireframe in Figura 3.7(a) rappresenta la schermata del login. L'utente ha la possibilità di inserire i propri dati o di registrarsi, nel caso in cui non l'abbia già fatto.

Schermata della Registrazione

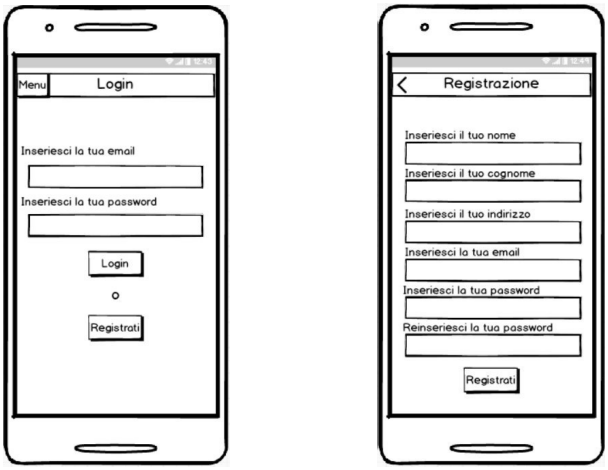
Il wireframe in Figura 3.7(b) rappresenta la schermata della registrazione. L'utente dovrà inserire i vari dati affinchè essa avvenga correttamente.

Schermata Profilo

Il wireframe in Figura 3.8(a) rappresenta la schermata del profilo di un utente.

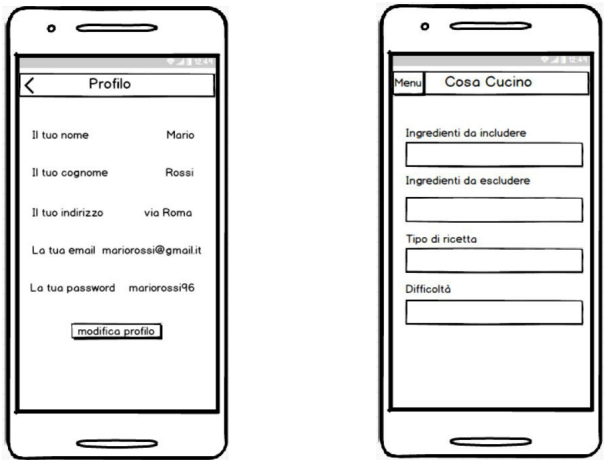
Schermata Cosa cucino

Il wireframe in Figura 3.8(b) rappresenta la schermata per filtrare, tramite alcuni criteri, le ricette presenti nella home page, in particolare quali ingredienti saranno inclusi o esclusi dalla ricerca, la tipologia della ricetta e la difficoltà che essa dovrà avere.



(a) *"Login"* (b) *"Registrazione"*

Figura 3.7: Wireframe delle schermate Login e Registrazione



(a) *"Profilo"* (b) *"Cosa Cucino"*

Figura 3.8: Wireframe delle schermate Profilo e Filtro

4

Implementazione e manuale utente

In questo capitolo verrà trattata la fase di implementazione per realizzare concretamente l'applicazione EAToday, tramite lo sviluppo del back-end su Altervista e dell'app su Android Studio. Infine, attraverso delle immagini, sarà stilato un piccolo manuale utente che illustrerà le principali funzionalità dell'applicazione.

4.1 Implementazione

L'implementazione dell'app può essere, essenzialemente, divisa in due sezioni, che verranno trattate in questo capitolo. Per prima cosa analizzeremo gli *Script PHP* utilizzati. Questi sono importanti in quanto permettono lo scambio di dati tra il database condiviso e l'applicazione, e viceversa, tramite l'uso di richieste *GET* e *POST*. In seguito, mostreremo le classi che sono state implementate per la realizzazione del progetto.

4.1.1 Script PHP

PHP, acronimo ricorsivo di *"PHP: Hypertext Preprocessor"*, ovvero preprocessore di ipertesti, è un linguaggio di scripting interpretato, utilizzato originariamente per la programmazione di pagine web dinamiche. In EAToday, gli script PHP vengono utilizzati per gestire sia le richieste dell'utente effettuate tramite l'app, sia il reperimento delle informazioni dal database condiviso. In poche parole, gli script svolgono il ruolo di *REST API*.

Script per filtrare le ricette

Il primo script PHP che analizziamo è mostrato nel Listato 4.1. Questo ha la funzione di caricare nella home page tutte le ricette presenti nel database, in modo tale che l'utente le possa consultare. Se, però, è presente una richiesta *GET* basata su dei parametri, lo script filtra le ricette sulla base della coppia chiave-valore, mostrando all'utente solo quelle ricette che rispettano le indicazioni fornite. In particolare, le ricette in EAToday possono essere filtrate dichiarando un alimento che

la ricetta deve contenere, oppure un alimento che non deve essere presente, e questa funzionalità di filtraggio viene eseguita dallo script del Listato 4.2. Le ricette possono essere filtrate anche sulla base di altri parametri. Infatti si può specificare che si sta cercando una pietanza per vegetariani o per vegani e, infine, si può scegliere la difficoltà della ricetta stessa.

```
1 $recipe_name = urldecode($_GET['name']);
2 $type = $_GET['type'];
3 $difficulty = $_GET['difficulty'];
4 $ingredients_s = $_GET['selected'];
5 $ingredients_e = $_GET['excluded'];
6 require "../connection.php";
7 header('Content-Type:Application/json');
8
9 function query_from_string($query, $con){
10 $result_recipes = mysqli_query($con,$query);
11     while ($row = mysqli_fetch_assoc($result_recipes)) {
12     $array_recipes[] = $row;
13     }
14     foreach ( $array_recipes as &$recipe){
15         $query_join = "SELECT * FROM collection INNER JOIN ingredient ON
                collection.id_ingredient = ingredient.id
16         WHERE collection.id_recipe = '".$recipe['id']."';";
17         $result_collection = mysqli_query($con,$query_join);
18         while ($row = mysqli_fetch_assoc($result_collection)) {
19         $array_ingredients[] = $row;
20       }
21         $recipe['ingredients'] = $array_ingredients;
22         unset($array_ingredients);
23     }
24     return $array_recipes;
25 }
```

Listato 4.1: Richiesta *GET* per inviare le ricette

```
1 $check = 0;
2 if ($ingredients_s != null){
3   $array = $array_recipes;
4   unset($array_recipes);
5   $ingredients_selected = explode(",",$ingredients_s);
6     foreach ( $array as $recipe){
7       $ingredients = $recipe['ingredients'];
8         foreach( $ingredients as $in){
9       $array_tmp[] = $in['id'];
10        }
11        foreach( $ingredients_selected as $in){
12      if (!in_array($in,$array_tmp)){
13          $check = 1;
14               break;
15          }
16        }
17        if ($check == 0){
18          $array_recipes[] = $recipe;
19          }
20        unset($array_tmp);
21        $check = 0;
22        }
23     unset($array);
24 }
25 echo json_encode($array_recipes);
```

Listato 4.2: Filtraggio degli ingredienti desiderati

Script per l'accesso

Gli script mostrati in questa regione gestiscono la rischiesta di accesso dell'utente. In particolare, lo script nel Listato 4.3 gestisce la richiesta di autenticazione. Esso richiede all'utente di inserire username e password per verificare la sua identità, e manda un messaggio di errore nel caso in cui il login non sia andato a buon fine.

Lo script nel Listato 4.4 gestisce, invece, la richiesta di registrazione del nuovo utente. In poche parole, esso richiede di inserire i propri dati personali per poter effettuare l'iscrizione e manda un messaggio di errore nel caso in cui un utente tenta di registrarsi con un'email che è già stata utilizzata.

```php
<?php
$json = file_get_contents('php://input');
$data = json_decode($json);
$name = $data->name;
$lastName = $data->lastName;
$email = $data->email;
$password = $data->password;
$address = $data->address;
require "../connection.php";
$query = "select * from user where email= '$email';";
$result = mysqli_query($connection,$query);
if($email != null && $password != null && $name != null){
if (mysqli_num_rows($result) > 0){
  $response = array();
    $code = "register_false";
    $message = "User already Exist...";
    array_push($response,array("code"=>$code,"message"=>$message));
    echo json_encode(array("server_response"=>$response));
}
else{
$newquery = "insert into user values('".$name."','".$lastName."','".$email."','".
      $password."','".$address."')";
    $newresult = mysqli_query($connection,$newquery);
    if(!$newresult){
      $response = array();
      $code = "register_false";
      $message = "Some server error occurred... Try again.";
      array_push($response,array("code"=>$code,"message"=>$message));
      echo json_encode(array("server_response"=>$response));
    }
    else{
      $response = array();
      $code = "register_true";
      $message = "Registration success";
      array_push($response,array("code"=>$code,"message"=>$message));
      echo json_encode(array("server_response"=>$response));
    }
}
}else{
      $response = array();
      $code = "register_false";
      $message = "Empty field";
      array_push($response,array("code"=>$code,"message"=>$message));
      echo json_encode(array("server_response"=>$response));
}
mysqli_close($connection);
?>
```

Listato 4.3: Richiesta *POST* per inviare email e password per effettuare l'accesso

```php
<?php
$json = file_get_contents('php://input');
$data = json_decode($json);
```

```php
$name = $data->name;
$lastName = $data->lastName;
$email = $data->email;
$password = $data->password;
$address = $data->address;
require "../connection.php";
$query = "select * from user where email= '$email';";
$result = mysqli_query($connection,$query);
if($email != null && $password != null && $name != null){
if (mysqli_num_rows($result) > 0){
  $response = array();
    $code = "register_false";
    $message = "User already Exist...";
    array_push($response,array("code"=>$code,"message"=>$message));
    echo json_encode(array("server_response"=>$response));
}
else{
$newquery = "insert into user values('".$name."','".$lastName."','".$email."','".
      $password."','".$address."')";
    $newresult = mysqli_query($connection,$newquery);
    if(!$newresult){
      $response = array();
      $code = "register_false";
      $message = "Some server error occurred... Try again.";
      array_push($response,array("code"=>$code,"message"=>$message));
      echo json_encode(array("server_response"=>$response));
    }
    else{
      $response = array();
      $code = "register_true";
      $message = "Registration success";
      array_push($response,array("code"=>$code,"message"=>$message));
      echo json_encode(array("server_response"=>$response));
    }
}
}else{
      $response = array();
      $code = "register_false";
      $message = "Empty field";
      array_push($response,array("code"=>$code,"message"=>$message));
      echo json_encode(array("server_response"=>$response));
}
mysqli_close($connection);
?>
```

Listato 4.4: Richiesta *POST* per inviare i dati utili per effettuare la registrazione

4.1.2 Definizione dei modelli principali

In questa sezione vengono illustrate le classi Java che sono state create per incapsulare i dati utili all'applicazione in formato *JSON*.

La classe del Listato 4.5 contiene tutti gli ingredienti, con relativi attributi, tra i quali codice identificativo, nome, disponibilità, prezzo e negozio. Quando un utente clicca su una ricetta per leggerne i dettagli, tramite i metodi di questa classe vengono visualizzati gli ingredienti che servono per la preparazione, con i relativi dettagli. Si pone l'attenzione, in particolare, al metodo `setPriceValue(String price)`, che ha lo scopo di convertire il prezzo del prodotto, estrapolato dal file *JSON*, in un numero decimale con precisione centesimale e di affiancargli il simbolo "€". Inoltre, il metodo `getDescriptionToString()` stampa sulla schermata il prezzo per unità di prodotto e il negozio preso il quale è disponibile.

```
1 public class Ingredient {
2
3     private int id;
4     private String name;
5     private String unit;
6     private String availability;
7     private String price;
8     private String store;
9
10    public Ingredient(String id, String name, String unit, String availability,
          String price, String store) {
11        this.id = Integer.parseInt(id);
12        this.name = name;
13        this.unit = unit;
14        this.availability = availability;
15        this.price = setPriceValue(price);
16        this.store = store;
17    }
18
19    public int getId() {
20        return id;
21    }
22
23    public void setId(int id) {
24        this.id = id;
25    }
26
27    public String getName() {
28        return name;
29    }
30
31    public void setName(String name) {
32        this.name = name;
33    }
34
35    public String getUnit() {
36        return unit;
37    }
38
39    public void setUnit(String unit) {
40        this.unit = unit;
41    }
42
43    public String getAvailability() {
44        return availability;
45    }
46
47    public void setAvailability(String availability) {
48        this.availability = availability;
49    }
50
51    public String getPrice() {
52        return price;
53    }
54
55    public void setPrice(String price) {
56        this.price = price;
57    }
58
59    private String setPriceValue(String price) {
60        float p = Float.parseFloat(price);
61        DecimalFormat decimalFormat = new DecimalFormat("0.00");
62        return decimalFormat.format(p) + " ?";
63    }
64
65    public String getStore() {
66        return store;
67    }
68
69    public void setStore(String store) {
```

```
70          this.store = store;
71      }
72
73      @Override
74      public String toString() {
75          return name + " " + store;
76      }
77
78      public String getDescriptionToString(){
79          return "Al costo di " + price + "\nper "+ unit + " di " + name + "\npresso
                il " + store;
80      }
81  }
```

Listato 4.5: Classe per incapsulare gli ingredienti

La classe del Listato 4.6 serve, invece, per incapsulare le ricette. Ogni ricetta ha degli attributi, tra cui un numero identificativo, il nome, il tempo di preparazione, la difficoltà, il prezzo per comprare tutti gli ingredienti necessari, la descrizione che spiega come realizzarla, l'URL dell'immagine dell'anteprima e un array che contiene tutti gli ingredienti. Tale classe è composta da costruttori e da metodi get e set per recuperare tutti i dati necessari. Un metodo caratteristico di questa classe è `setPrice(ArrayList<Ingredient> ingredients)` che calcola il costo totale della spesa per preparare la ricetta, sommando i prezzi dei singoli ingredienti. Questo numero viene, poi, convertito in numero decimale con precisione centesimale, e gli viene affiancato il simbolo "€". Un altro metodo importante è `createImageUrlByName(String name)`, che ha l'obiettivo di comporre l'URL dell'immagine da associare alla ricetta. Il metodo riceve in ingresso il nome della ricetta, sostituisce gli spazi vuoti con il carattere "_" e aggiunge "`.png`". Tale URL verrà, poi, inviato per ricevere l'immagine da visualizzare.

```
 1  public class Recipe {
 2      private static int counter;
 3      private int id;
 4      private String name;
 5      private String time;
 6      private String difficulty;
 7      private String price;
 8      private String type;
 9      private String description;
10      private String imageUrl;
11      private ArrayList<Ingredient> ingredients;
12
13      public Recipe(String name, String time, String difficulty, String type, String
              description, ArrayList<Ingredient> ingredients ) throws
            UnsupportedEncodingException {
14          this.id = setId();
15          this.name = name;
16          this.time = time + " min";
17          this.difficulty = difficulty;
18          this.type = type;
19          this.description = description;
20          this.imageUrl = createImageUrlByName(name);
21          this.ingredients = ingredients;
22          this.price = setPrice(ingredients);
23      }
24
25      public int getId() {
26          return id;
27      }
28
29      private int setId() {
30          this.id = counter;
```

```
31          counter++;
32          return this.id;
33      }
34
35      public String getName() {
36          return name;
37      }
38
39      public void setName(String name) {
40          this.name = name;
41      }
42
43      public String getTime() {
44          return time;
45      }
46
47      public void setTime(String time) {
48          this.time = time;
49      }
50
51      public String getDifficulty() {
52          return difficulty;
53      }
54
55      public void setDifficulty(String difficulty) {
56          this.difficulty = difficulty;
57      }
58
59      public String getPrice() {
60          return price;
61      }
62
63      public void setPrice(String price) {
64          this.price = price;
65      }
66
67      private String setPrice(ArrayList<Ingredient> ingredients){
68          float sum = 0;
69          for (int i=0; i < ingredients.size(); i++){
70              String s = ingredients.get(i).getPrice().replaceAll("?","").trim();
71              sum += Float.parseFloat(s);
72          }
73          DecimalFormat decimalFormat = new DecimalFormat("0.00");
74          return decimalFormat.format(sum) + " ?";
75      }
76
77      public String getType() {
78          return type;
79      }
80
81      public void setType(String type) {
82          this.type = type;
83      }
84
85      public String getDescription() {
86          return description;
87      }
88
89      public void setDescription(String description) {
90          this.description = description;
91      }
92
93      public String getImageUrl() {
94          return imageUrl;
95      }
96
97      public void setImageUrl(String imageUrl) {
98          this.imageUrl = imageUrl;
99      }
100
```

```
101     public ArrayList<Ingredient> getIngredients() {
102         return ingredients;
103     }
104
105     public void setIngredients(ArrayList<Ingredient> ingredients) {
106         this.ingredients = ingredients;
107     }
108
109     private String createImageUrlByName(String name){
110         String url = Constant.IMAGES_PATH + name.replaceAll(" ","_") + Constant.
                PNG;
111         return url;
112     }
113 }
```

Listato 4.6: Classe per incapsulare le ricette

La classe nel Listato 4.7 è utile per incapsulare i dati dell'utente. In particolare, contiene al proprio interno nome, cognome, email, password, indirizzo e un attributo booleano che indica se l'utente ha effettuato il login. Questa classe contiene solo il costrutture e i metodi get e set per il recupero delle informazioni.

```
 1 public class User {
 2     private static String name;
 3     private static String lastName;
 4     private static String email;
 5     private static String password;
 6     private static String address;
 7     private static Boolean isLog = false;
 8
 9     public static String getName() {
10         return name;
11     }
12
13     public static void setName(String name) {
14         User.name = name;
15     }
16
17     public static String getLastName() {
18         return lastName;
19     }
20
21     public static void setLastName(String lastName) {
22         User.lastName = lastName;
23     }
24
25     public static String getEmail() {
26         return email;
27     }
28
29     public static void setEmail(String email) {
30         User.email = email;
31     }
32
33     public static String getPassword() {
34         return password;
35     }
36
37     public static void setPassword(String password) {
38         User.password = password;
39     }
40
41     public static String getAddress() {
42         return address;
43     }
44
45     public static void setAddress(String address) {
46         User.address = address;
```

```
47     }
48
49     public static Boolean getIsLog() {
50         return isLog;
51     }
52
53     public static void setIsLog(Boolean isLog) {
54         User.isLog = isLog;
55     }
56 }
```

Listato 4.7: Classe per incapsulare i dati dell'utente

4.1.3 Schermata iniziale dell'app

Con schermata iniziale dell'app s'indende la `MainActivity` che compare all'apertura dell'applicazione. Per prima cosa, vengono scaricate dal database tutte le ricette disponibili così da poterle mostrare all'utente appena egli è entrato nell'app; tali ricette verranno disposte una sotto l'altra, e verranno visualizzate delle foto d'anteprima, il nome e le caratteristiche che le contraddistinguono, come la difficoltà e tempo di preparazione. Inoltre, all'apertura dell'app, viene effettuato un controllo delle credenziali d'accesso dell'utente; se queste sono salvate nell'applicazione, allora verrà automaticamente eseguito il login.

Drawer menù

Il drawer menù, mostrato nel Listato 4.8, è posizionato in alto a sinistra nella schermata iniziale. Una volta che l'utente ha premuto sul pulsante corrispondente, si aprirà questo menù che contiene diverse voci, che differiscono a seconda che l'utente abbia già effettuato l'accesso oppure no. In quest'ultimo caso le opzioni presenti nel menù sono "home", "login", "filtra ricette" e "progetto". Se, invece, l'utente ha effettuato l'autenticazione, al posto di "login" sarà presente la voce "profilo", per accedere ai propri dati personali. Inoltre ci sono due opzioni supplementari, ovvero "le tue ricette", per vedere le ricette che sono state salvate, e "logout", per disconnettersi dall'app.

```
1 private void initToolbar() {
2         toolbar = this.findViewById(R.id.toolbar);
3         drawerLayout = this.findViewById(R.id.drawer_layout);
4         navigationView = this.findViewById(R.id.navigationView);
5         this.setSupportActionBar(toolbar);
6
7         final ActionBar actionBar = this.getSupportActionBar();
8         assert actionBar != null;
9         actionBar.setDisplayHomeAsUpEnabled(true);
10        actionBar.setHomeAsUpIndicator(R.drawable.ic_menu);
11
12        if (User.getIsLog()){
13            navigationView.getMenu().findItem(R.id.nav2).setTitle(R.string.menu_21)
                  ;
14            navigationView.getMenu().findItem(R.id.nav2).setIcon(R.drawable.ic_user
                  );
15        }else{
16            navigationView.getMenu().findItem(R.id.nav5).setVisible(false);
17            navigationView.getMenu().findItem(R.id.nav6).setVisible(false);
18        }
```

```
19
20        navigationView.setNavigationItemSelectedListener(new NavigationView.
              OnNavigationItemSelectedListener() {
21
22            @Override
23            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
24
25                switch (menuItem.getItemId()){
26                    case R.id.nav1:
27                        menuItem.setChecked(true);
28                        drawerLayout.closeDrawers();
29                        return true;
30                    case R.id.nav2:
31                        if (User.getIsLog()){
32                            Intent intent = new Intent(getApplicationContext(),
                                  ProfileActivity.class);
33                            startActivity(intent);
34                        }else{
35                            menuItem.setChecked(true);
36                            Intent intent = new Intent(getApplicationContext(),
                                  LoginActivity.class);
37                            startActivity(intent);
38                        }
39                        drawerLayout.closeDrawers();
40                        return true;
41                    case R.id.nav3:
42                        menuItem.setChecked(true);
43                        menuItem.setChecked(true);
44                        Intent intentF = new Intent(getApplicationContext(),
                              FilterActivity.class);
45                        startActivity(intentF);
46                        drawerLayout.closeDrawers();
47                        return true;
48                    case R.id.nav4:
49                        menuItem.setChecked(true);
50                        Intent intentP = new Intent(getApplicationContext(),
                              ProjectActivity.class);
51                        startActivity(intentP);
52                        drawerLayout.closeDrawers();
53                        return true;
54                    case R.id.nav5:
55                        menuItem.setChecked(true);
56                        AlertDialog.Builder builder = new AlertDialog.Builder (
                              MainActivity.this);
57                        builder.setTitle("Logout");
58                        builder.setMessage("Sei sicuro di voler uscire?");
59                        builder.setPositiveButton("Si", new DialogInterface.
                              OnClickListener() {
60                            @Override
61                            public void onClick(DialogInterface dialogInterface, int
                                  i) {
62                                dialogInterface.dismiss();
63                                PreferenceUtils.logout(getApplicationContext());
64                                Intent intentM = new Intent(getApplicationContext(),
                                      MainActivity.class);
65                                startActivity(intentM);
66                            }
67                        });
68                        AlertDialog alertDialog = builder.create();
69                        alertDialog.show();
70                        drawerLayout.closeDrawers();
71                        return true;
72                    case R.id.nav6:
73                        menuItem.setChecked(true);
74                        Intent intentP = new Intent(getApplicationContext(),
                              RecipeSavedActivity.class);
75                        startActivity(intentP);
76                        drawerLayout.closeDrawers();
77                        return true;
78                }
```

```
79              return false;
80          }
81
82      });
83  }
```

Listato 4.8: Metodo della classe `MainActivity` richiamato nel metodo `OnCreate()` per inizializzare il drawer menù

Visualizzazione delle ricette

La visualizzazione delle ricette all'interno della `MainActivity` è controllata da tre classi differenti. La prima è quella mostrata nel Listato 4.9. Questa, inizialmente, controlla se sono state effettuate delle richieste di tipo *GET* dalla sezione "filtra ricette", chiama la classe `RecipeLoader` (mostrata nel Listato 4.10), passa ad essa come stringa le eventuali richieste e poi si mette in attesa. Questa classe appena chiamata scarica dal database le ricette disponibili, combinandole con le richieste *GET* presenti, e crea un `arrayList` in cui carica le ricette che corrispondono ai requisiti richiesti dall'utente. Questo `arrayList` viene inviato alla prima classe, che chiama, necessariamente, la classe `RecipeAdapter` (Listato 4.11), passando ad essa come argomento tale `arrayList` di ricette. `RecipeAdapter` ha, infine, lo scopo di prendere i dati di ogni ricetta e di inserirli nella `RecyclerView`, così che le ricette stesse possano essere visualizzate dall'utente.

```
1       recyclerViewRecipe = findViewById(R.id.list_details);
2       recyclerViewRecipe.setHasFixedSize(true);
3       layoutManagerRecipe = new LinearLayoutManager(this);
4       recyclerViewRecipe.setLayoutManager(layoutManagerRecipe);
5       Intent intent = getIntent();
6       String get = "";
7       if (intent.hasExtra("getRequest")){
8           get = getIntent().getStringExtra("getRequest");
9       }
10      CountDownLatch latch = new CountDownLatch(1);
11      RecipeLoader recipeLoader = new RecipeLoader((Context) MainActivity.this,
            latch);
12      recipeLoader.execute(get);
13      try {
14          latch.await();
15      } catch (InterruptedException e) {
16          e.printStackTrace();
17      }
18      myRecipeAdapter = new RecipeAdapter(this,(Context) MainActivity.this,
            RecipeCollection.recipesList);
19      recyclerViewRecipe.setAdapter(myRecipeAdapter);
20      if (!intent.hasExtra("getRequest")){
21          RecipeCollection.startRecipesList = RecipeCollection.recipesList;
22      }
23      CountDownLatch latch1 = new CountDownLatch(1);
24      IngredientLoader ingredientLoader = new IngredientLoader((Context)
            MainActivity.this, latch1);
25      ingredientLoader.execute();
26      try {
27          latch1.await();
28      } catch (InterruptedException e) {
29          e.printStackTrace();
30      }
```

Listato 4.9: Processo all'interno del metodo `OnCreate()` della `MainActivity` per inizializzare la `RecyclerView` di ricette

```java
public class RecipeLoader extends AsyncTask<String, Void, String> {

    private Context context;
    private Activity activity;
    private AlertDialog.Builder builder;
    private ProgressDialog progressDialog;
    public ArrayList<Recipe> arrayList = new ArrayList<>();
    private CountDownLatch latch;
    private StringBuilder stringBuilder;

    public RecipeLoader(Context context, CountDownLatch latch) {
        this.context = context;
        this.activity = (Activity) context;
        this.latch = latch;
    }
    @Override
    protected void onPreExecute() {
        builder = new AlertDialog.Builder(activity);
        progressDialog = new ProgressDialog(context);
        progressDialog.setTitle("Please wait...");
        progressDialog.setMessage("Connecting to server");
        progressDialog.setIndeterminate(true);
        progressDialog.setCancelable(false);
        progressDialog.show();
    }
    @Override
    protected String doInBackground(String... getString) {
        try {
            URL url = new URL(Constant.URL_RECIPE + getString[0]);
            String json = connectionResult(url);
            RecipeCollection.recipesList = arrayList;
            latch.countDown();
            return json;
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        return null;
    }
    private String connectionResult(URL url) {
        try {
            HttpURLConnection httpURLConnection = (HttpURLConnection) url.
                 openConnection();
            httpURLConnection.setRequestMethod("GET");
            httpURLConnection.setRequestProperty("Content-Type", "application/json"
                 );
            httpURLConnection.setRequestProperty("Accept", "application/json");
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setRequestProperty("Connection", "Keep-Alive");
            httpURLConnection.setDoInput(true);
            httpURLConnection.connect();

            int code = httpURLConnection.getResponseCode();
            if (code == 200) {
                InputStream inputStream = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new
                     InputStreamReader(inputStream));
                stringBuilder = new StringBuilder();
                String line = "";
                while ((line = bufferedReader.readLine()) != null) {
                    stringBuilder.append(line);
                }
                Thread.sleep(2000);
                httpURLConnection.disconnect();
                String json = stringBuilder.toString().trim();
                progressDialog.dismiss();
                serializeJson(json);
                return json;
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
68            AlertDialog.Builder builder = new AlertDialog.Builder (context);
69            builder.setTitle("Connection failed");
70            builder.setMessage("Please check your connection");
71            builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
72                @Override
73                public void onClick(DialogInterface dialogInterface, int i) {
74                    dialogInterface.dismiss();
75                }
76            });
77            AlertDialog alertDialog = builder.create();
78            alertDialog.show();
79            try {
80                Thread.sleep(3000);
81            } catch (InterruptedException ex) {
82                ex.printStackTrace();
83            }
84        }
85        return null;
86    }
87    private void serializeJson(String json) throws UnsupportedEncodingException {
88        try {
89            JSONArray jsonArray = new JSONArray(json);
90            for(int i=0; i < jsonArray.length(); i++){
91                JSONArray ingredients = jsonArray.getJSONObject(i).getJSONArray("
                      ingredients");
92                ArrayList<Ingredient> ingredientArrayList = new ArrayList<>();
93                for( int j = 0; j < ingredients.length(); j++){
94                    ingredientArrayList.add(new Ingredient(ingredients.
                          getJSONObject(j).getString("id"),
95                            ingredients.getJSONObject(j).getString("name"),
96                            ingredients.getJSONObject(j).getString("unit"),
97                            ingredients.getJSONObject(j).getString("availability"),
98                            ingredients.getJSONObject(j).getString("price"),
99                            ingredients.getJSONObject(j).getString("store")));
100               }
101               arrayList.add(new Recipe(jsonArray.getJSONObject(i).getString("name
                      "),
102                       jsonArray.getJSONObject(i).getString("time"),
103                       jsonArray.getJSONObject(i).getString("difficulty"),
104                       jsonArray.getJSONObject(i).getString("type"),
105                       jsonArray.getJSONObject(i).getString("description"),
106                       ingredientArrayList));
107           }
108       } catch (JSONException ex) {
109           ex.printStackTrace();
110       }
111   }
112 }
```

Listato 4.10: Classe per caricare le ricette contenute nel database

```
1
2 public class RecipeAdapter extends RecyclerView.Adapter<RecipeAdapter.ViewHolder>
      {
3
4     private ArrayList<Recipe> recipes;
5     private ItemClicked activity;
6     private Context context;
7     private Activity thisClass;
8
9     public RecipeAdapter(Activity thisClass, Context context, ArrayList<Recipe>
          list){
10        this.recipes = list;
11        this.activity = (ItemClicked) context;
12        this.context = context;
13        this.thisClass = thisClass;
14    }
15    public interface ItemClicked{
16        void onItemClicked(int index);
```

```
17    }
18    public class ViewHolder extends RecyclerView.ViewHolder{
19
20        private ImageView ivImageRecipe;
21        private ImageView imageVTime;
22        private ImageView imageVPrice;
23        private ImageView imageVDifficulty;
24        private ImageView imageVType;
25        private TextView tvRecipeName;
26        private TextView tvRecipeTime;
27        private TextView tvRecipeDifficulty;
28        private TextView tvRecipePrice;
29        private TextView tvRecipeType;
30        private LinearLayout linearLayoutTop;
31        private LinearLayout linearLayoutBottom;
32
33        public ViewHolder(@NonNull View itemView) {
34            super(itemView);
35            ivImageRecipe = itemView.findViewById(R.id.ivImageRecipe);
36            imageVTime = itemView.findViewById(R.id.imageVTime);
37            imageVPrice = itemView.findViewById(R.id.imageVPrice);
38            imageVDifficulty = itemView.findViewById(R.id.imageVDifficulty);
39            imageVType = itemView.findViewById(R.id.imageVType);
40            tvRecipeName = itemView.findViewById(R.id.tvRecipeName);
41            tvRecipeTime = itemView.findViewById(R.id.tvRecipeTime);
42            tvRecipeDifficulty = itemView.findViewById(R.id.tvRecipeDifficulty);
43            tvRecipePrice = itemView.findViewById(R.id.tvRecipePrice);
44            tvRecipeType = itemView.findViewById(R.id.tvRecipeType);
45            linearLayoutTop = itemView.findViewById(R.id.layout_top);
46            linearLayoutBottom = itemView.findViewById(R.id.layout_bottom);
47
48
49            itemView.setOnClickListener(new View.OnClickListener() {
50                @Override
51                public void onClick(View view) {
52                    activity.onItemClicked(recipes.indexOf((Recipe) view.getTag()))
                          ;
53                }
54            });
55        }
56    }
57    @NonNull
58    @Override
59    public RecipeAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
          int viewType) {
60        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.
              recipe_list_item, parent,false);
61        return new ViewHolder(v);
62    }
63    @Override
64    public void onBindViewHolder(@NonNull RecipeAdapter.ViewHolder holder, int
          position) {
65        Recipe recipe = recipes.get(position);
66        holder.itemView.setTag(recipe);
67        holder.tvRecipeName.setText(recipes.get(position).getName());
68        Glide.with(thisClass.getApplicationContext())
69                .load(recipe.getImageUrl())
70                .placeholder(R.drawable.ic_broken_image_black_24dp)
71                .into(holder.ivImageRecipe);
72        holder.tvRecipeTime.setText(recipe.getTime());
73        holder.tvRecipeDifficulty.setText(recipe.getDifficulty());
74        holder.tvRecipePrice.setText(recipe.getPrice());
75        holder.tvRecipeType.setText(recipe.getType());
76        ViewGroup.LayoutParams paramsTop = holder.linearLayoutTop.getLayoutParams
              ();
77        paramsTop.height = 120;
78        holder.linearLayoutTop.setLayoutParams(paramsTop);
79        ViewGroup.LayoutParams paramsBottom = holder.linearLayoutBottom.
              getLayoutParams();
80        paramsBottom.height = 120;
```

```
81          holder.linearLayoutBottom.setLayoutParams(paramsBottom);
82
83          if(position == 0){
84              holder.itemView.setPadding(0,0,0,0);
85          }
86          else if(position == (recipes.size() -1)){
87              holder.itemView.setPadding(0,0,0,200);
88          }
89      }
90      @Override
91      public int getItemCount() {
92          return recipes.size();
93      }
94  }
```

Listato 4.11: Classe `RecipeAdapter` per visualizzare ogni singola ricetta

Barra di ricerca

La barra di ricerca, il cui codice è mostrato nel Listato 4.12, si trova nella `MainActivity`. Lo scopo di questa barra è quello di permettere all'utente di visualizzare una determinata ricetta, scrivendone la prima parte del nome, per facilitare la ricerca all'interno della schermata.

```
1  private void initSearchView(){
2          searchView = findViewById(R.id.search_by_name);
3          searchView.setQueryHint("Cerca tra le ricette...");
4          searchView.setBackgroundResource(R.drawable.search);
5          arrayListOnSearch = new ArrayList<>();
6          searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
7              @Override
8              public boolean onQueryTextSubmit(String text) {
9                  if(text.equals(""))RecipeCollection.recipesList = RecipeCollection.
                       startRecipesList;
10                 ArrayList<Recipe> arrayList = RecipeCollection.searchRecipesByName(
                       text);
11                 if (!arrayListOnSearch.equals(arrayList)){
12                     myRecipeAdapter = new RecipeAdapter(MainActivity.this,(Context)
                           MainActivity.this,arrayList);
13                     recyclerViewRecipe.setAdapter(myRecipeAdapter);
14                     myRecipeAdapter.notifyDataSetChanged();
15                     arrayListOnSearch.clear();
16                     arrayListOnSearch = arrayList;
17                     RecipeCollection.recipesList=arrayListOnSearch;
18                 }
19                 return false;
20             }
21
22             @Override
23             public boolean onQueryTextChange(String text) {
24                 if(text.equals(""))RecipeCollection.recipesList = RecipeCollection.
                       startRecipesList;
25                 ArrayList<Recipe> arrayList = RecipeCollection.searchRecipesByName(
                       text);
26                 if (!arrayListOnSearch.equals(arrayList)){
27                     myRecipeAdapter = new RecipeAdapter(MainActivity.this,(Context)
                           MainActivity.this,arrayList);
28                     recyclerViewRecipe.setAdapter(myRecipeAdapter);
29                     myRecipeAdapter.notifyDataSetChanged();
30                     arrayListOnSearch.clear();
31                     arrayListOnSearch = arrayList;
32                     RecipeCollection.recipesList=arrayListOnSearch;
33                 }
34                 return false;
35             }
```

```
36        });
37    }
```

Listato 4.12: Metodo della classe `MainActivity`, richiamato nel metodo `OnCreate()`, per inizializzare la barra di ricerca

4.1.4 Filtro ricette

Quando dal menù l'utente preme sull'opzione "filtra ricette" si apre una nuova `activity` che permette di inserire parametri specifici per la ricerca delle ricette. Tali parametri sono inizializzati tramite il codice mostrato nel Listato 4.13. L'utente ha la possibilità di specificare degli ingredienti che devono far parte della ricetta o degli ingredienti che non devono essere presenti. Per far ciò, vengono inizializzati due `arrayList` che conterranno, rispettivamente, gli alimenti che devono essere presenti e quelli che devono essere esclusi. L'utente può, poi, stabilire che la ricetta deve essere per vegetariani o per vegani, e ciò è possibile tramite due pulsanti switch. Inifine l'utente può selezionare la difficoltà della ricetta e, per questo, viene inizializzata una seekbar. Il processo interno al metodo `btnSearch.setOnClickListener()` è particolarmente importante in quanto riceve tutti i paramentri inseriti precedentemente e li usa per comporre la richiesta di tipo *GET* da inviare. Le ricette che coincidono con i paramentri inseriti saranno visualizzate nella home page. Si pone l'attenzione anche sulla funzione descritta nel Listato 4.14. Questa serve per poter eliminare gli alimenti nei campi degli ingredienti da includere o escludere che sono stati selezionati erroneamente. Per rendere più usabile l'applicazione, tramite questa funzione, si possono eliminare gli ingredienti semplicemente con uno swipe, verso destra o verso sinistra.

```
1        ids = new ArrayList<>();
2        idOK = new ArrayList<>();
3        idNO = new ArrayList<>();
4        listIngredientsOK = new ArrayList<>();
5        listIngredientsNO = new ArrayList<>();
6
7        btnOK.setOnClickListener(new View.OnClickListener() {
8            @Override
9            public void onClick(View view) {
10               String ingredientName = ingredientOK.getText().toString().trim();
11               ArrayList<String> list = new ArrayList<String>();
12               for(Ingredient in : RecipeCollection.ingredientsList){
13                   if(in.toString().contains(ingredientName)){
14                       list.add(in.getName());
15                       ids.add(in.getId());
16                   }
17               }
18
19               final ArrayAdapter< String > adapter = new ArrayAdapter < String >
20                       (FilterActivity.this, android.R.layout.simple_list_item_1,
21                               list);
22
23               AlertDialog.Builder builder=new AlertDialog.Builder(FilterActivity.
                     this);
24               if(ids.isEmpty()){
25                   Toast.makeText(FilterActivity.this,"Nessuno ingrediente trovato
                         ",Toast.LENGTH_SHORT).show();
26               }else{
27                   builder.setTitle("Ingredienti trovati");
28                   builder.setMessage("Selezionare l'ingrediente corretto");
```

```
29                  DialogInterface.OnClickListener listener = new DialogInterface.
                        OnClickListener() {
30                      @Override
31                      public void onClick(DialogInterface dialogInterface, int i)
                            {
32                          String add = adapter.getItem(i);
33                          int id = ids.get(i);
34                          idOK.add(id);
35                          ids.clear();
36                          listIngredientsOK.add(add);
37                          addToListView(listViewOK, listIngredientsOK);
38                      }
39                  };
40                  builder.setAdapter(adapter,listener);
41                  AlertDialog alertDialog = builder.create();
42                  ListView listView = alertDialog.getListView();
43                  listView.setEnabled(true);
44                  alertDialog.setView(listView);
45                  alertDialog.show();
46              }
47          }
48      });
49
50      btnNO.setOnClickListener(new View.OnClickListener() {
51          @Override
52          public void onClick(View view) {
53              String ingredientName = ingredientNO.getText().toString().trim();
54              ArrayList<String> list = new ArrayList<String>();
55              for(Ingredient in : RecipeCollection.ingredientsList){
56                  if(in.toString().contains(ingredientName)){
57                      list.add(in.getName());
58                      ids.add(in.getId());
59                  }
60              }
61
62              final ArrayAdapter< String > adapter = new ArrayAdapter < String >
63                      (FilterActivity.this, android.R.layout.simple_list_item_1,
64                              list);
65
66              AlertDialog.Builder builder=new AlertDialog.Builder(FilterActivity.
                        this);
67              if(ids.isEmpty()){
68                  Toast.makeText(FilterActivity.this,"Nessuno ingrediente trovato
                            ",Toast.LENGTH_SHORT).show();
69              }else{
70                  builder.setTitle("Ingredienti trovati");
71                  builder.setMessage("Selezionare l'ingrediente corretto");
72                  DialogInterface.OnClickListener listener = new DialogInterface.
                        OnClickListener() {
73                      @Override
74                      public void onClick(DialogInterface dialogInterface, int i)
                            {
75                          String add = adapter.getItem(i);
76                          int id = ids.get(i);
77                          idNO.add(id);
78                          ids.clear();
79                          listIngredientsNO.add(add);
80                          addToListView(listViewNO, listIngredientsNO);
81                      }
82                  };
83                  builder.setAdapter(adapter,listener);
84                  AlertDialog alertDialog = builder.create();
85                  ListView listView = alertDialog.getListView();
86                  listView.setEnabled(true);
87                  alertDialog.setView(listView);
88                  alertDialog.show();
89              }
90          }
91      });
92
```

```
93          vegetarian.setOnCheckedChangeListener(new CompoundButton.
                OnCheckedChangeListener() {
94              public void onCheckedChanged(CompoundButton buttonView, boolean
                     isChecked) {
95                  if(isChecked){
96                      isVegetarian = true;
97                      isVegan = true;
98                      vegan.setChecked(true);
99                  }else{
100                     isVegetarian = false;
101                 }
102             }
103         });
104
105         vegan.setOnCheckedChangeListener(new CompoundButton.
                OnCheckedChangeListener() {
106             public void onCheckedChanged(CompoundButton buttonView, boolean
                     isChecked) {
107                 if(isChecked){
108                     isVegan = true;
109                 }else{
110                     isVegan = false;
111                 }
112             }
113         });
114
115         seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
116             @Override
117             public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
118                 switch (i){
119                     case (1):
120                         difficulty = "facile";
121                         break;
122                     case (2):
123                         difficulty = "medio";
124                         break;
125                     case (3):
126                         difficulty = "difficile";
127                         break;
128                     default:
129                         difficulty = "";
130                 }
131             }
132
133             @Override
134             public void onStartTrackingTouch(SeekBar seekBar) {
135
136             }
137
138             @Override
139             public void onStopTrackingTouch(SeekBar seekBar) {
140
141             }
142         });
143
144         btnBack.setOnClickListener(new View.OnClickListener() {
145             @Override
146             public void onClick(View view) {
147                 onBackPressed();
148             }
149         });
150
151         btnSearch.setOnClickListener(new View.OnClickListener() {
152             @Override
153             public void onClick(View view) {
154                 Intent intent = new Intent(getApplicationContext(), MainActivity.
                        class);
155                 String get = "?";
156                 if(!idOK.isEmpty()){
157                     get += Constant.SELECTED;
```

```
158                 int size = idOK.size() - 1;
159                 for(int i= 0; i < size ; i++){
160                     get += String.valueOf(idOK.get(i));
161                     get += ",";
162                 }
163                 get += String.valueOf(idOK.get(size));
164             }
165             if(!idNO.isEmpty()){
166                 if(!get.equals("?")) get += "&";
167                 get += Constant.EXCLUDED;
168                 int size = idNO.size() - 1;
169                 for(int i= 0; i < size ; i++){
170                     get += String.valueOf(idNO.get(i));
171                     get += ",";
172                 }
173                 get += String.valueOf(idNO.get(size));
174             }
175             if(isVegetarian){
176                 if(!get.equals("?")) get += "&";
177                 get += Constant.TYPE + "vegetariano";
178             }else{
179                 if(isVegan){
180                     if(!get.equals("?")) get += "&";
181                     get += Constant.TYPE + "vegano";
182                 }
183             }
184             if(!difficulty.equals("")){
185                 if(!get.equals("?")) get += "&";
186                 get += Constant.DIFFICULTY + difficulty;
187             }
188             intent.putExtra("getRequest", get);
189             finish();
190             startActivity(intent);
191         }
192     });
193 }
194
195 private void addToListView(RecyclerView listView, ArrayList<String>
        listElements){
196
197     listView.setLayoutManager(new GridLayoutManager(FilterActivity.this, 1));
198     myAdapter = new StringAdapter(FilterActivity.this, FilterActivity.this,
            listElements);
199     listView.setAdapter(myAdapter);
200
201     itemTouchHelper.attachToRecyclerView(listView);
202     listView.setScrollContainer(false);
203
204     myAdapter.notifyDataSetChanged();
205     ingredientOK.getText().clear();
206     ingredientNO.getText().clear();
207 }
```

Listato 4.13: Inizializzazione dei componenti usati nella schermata Filtro

```
1 ItemTouchHelper itemTouchHelper = new ItemTouchHelper(new ItemTouchHelper.
      SimpleCallback(0, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
2       @Override
3       public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull
            RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder
            target) {
4           return false;
5       }
6
7       @Override
8       public void onSwiped(RecyclerView.ViewHolder viewHolder, int swipeDir) {
9
10          String name = myAdapter.getString(viewHolder.getAdapterPosition());
11          for(int i = 0; i < listIngredientsOK.size(); i++){
```

```
12              if(listIngredientsOK.get(i).equals(name)){
13                  listIngredientsOK.remove(i);
14                  idOK.remove(i);
15              }
16          }
17          for(int i = 0; i < listIngredientsNO.size(); i++){
18              if(listIngredientsNO.get(i).equals(name)){
19                  listIngredientsNO.remove(i);
20                  idNO.remove(i);
21              }
22          }
23          myAdapter.notifyDataSetChanged();
24      }
25  });
```

Listato 4.14: Componente istanziato per eliminare gli ingredienti in lista tramite uno swipe

4.1.5 Dettagli ricetta

La classe `DetailsActivity`, mostrata nel Listato 4.15, viene utilizzata per inizializzare tutti i dettagli relativi alle ricette e ai loro ingredienti. Si pone l'attenzione su `btnOrd.setOnClickListener()`, ovvero sul pulsante che serve per acquistare gli ingredienti della ricetta selezionata. Una volta premuto, la funzione controlla se l'utente ha eseguito il login; in tal caso si procede all'acquisto. In caso contrario, si viene indirizzati alla schermata per effettuare l'accesso.

```
1 public class DetailsActivity extends AppCompatActivity implements
      IngredientAdapter.ItemClicked {
2
3     private ImageView ivImageRecipe;
4     private TextView tvName;
5     private TextView description;
6     private Button btnBack;
7     private Button btnOrd;
8     private int index;
9     private RecyclerView recyclerViewIngredient;
10    private RecyclerView.Adapter myIngredientAdapter;
11    private RecyclerView.LayoutManager layoutManagerIngredient;
12
13
14    @Override
15    protected void onCreate(Bundle savedInstanceState) {
16        super.onCreate(savedInstanceState);
17        setContentView(R.layout.details_main);
18        setTitle(R.string.app_name);
19
20        ivImageRecipe = findViewById(R.id.imageViewOfRecipe);
21        tvName = findViewById(R.id.tvName);
22        description = findViewById(R.id.textarea);
23        btnBack = findViewById(R.id.btn_back);
24        btnOrd = findViewById(R.id.btn_ord);
25
26        ArrayList<Ingredient> ingredients = new ArrayList<>();
27        Intent intent = getIntent();
28        if (intent.hasExtra("recipeIndex")) {
29            index = getIntent().getIntExtra("recipeIndex",0);
30            tvName.setText(RecipeCollection.recipesList.get(index).getName());
31            Glide.with(this.getApplicationContext())
32                    .load(RecipeCollection.recipesList.get(index).getImageUrl())
33                    .placeholder(R.drawable.ic_broken_image_black_24dp)
34                    .into(ivImageRecipe);
35            description.setText(RecipeCollection.recipesList.get(index).
                  getDescription());
```

```
36            ingredients = RecipeCollection.recipesList.get(index).getIngredients();
37        }
38
39        recyclerViewIngredient = findViewById(R.id.list_ingredients);
40        recyclerViewIngredient.setHasFixedSize(true);
41        layoutManagerIngredient = new LinearLayoutManager(this);
42        recyclerViewIngredient.setLayoutManager(layoutManagerIngredient);
43        myIngredientAdapter = new IngredientAdapter(this,(Context) DetailsActivity
              .this, ingredients);
44        recyclerViewIngredient.setAdapter(myIngredientAdapter);
45
46        ViewGroup.LayoutParams layoutParams = ivImageRecipe.getLayoutParams();
47        DisplayMetrics displayMetrics = new DisplayMetrics();
48        getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
49        int heigh = (int) (displayMetrics.widthPixels*(0.45));
50        layoutParams.height = heigh;
51        ivImageRecipe.setLayoutParams(layoutParams);
52
53        btnOrd.setOnClickListener(new View.OnClickListener() {
54            @Override
55            public void onClick(View view) {
56                AlertDialog.Builder builder = new AlertDialog.Builder (
                      DetailsActivity.this);
57                if( User.getIsLog()){
58                    builder.setTitle("Ordine pronto");
59                    builder.setMessage("Grazie per aver utilizzato EAToday!");
60                    builder.setPositiveButton("Home", new DialogInterface.
                          OnClickListener() {
61                        @Override
62                        public void onClick(DialogInterface dialogInterface, int i)
                              {
63                            dialogInterface.dismiss();
64                            finish();
65                            Intent intent = new Intent(getApplicationContext(),
                                  MainActivity.class);
66                            startActivity(intent);
67                        }
68                    });
69                }else{
70                    builder.setTitle("Attezione");
71                    builder.setMessage("Per poter effettuare l'ordine bisogna
                          effettuare il login.");
72                    builder.setPositiveButton("Login", new DialogInterface.
                          OnClickListener() {
73                        @Override
74                        public void onClick(DialogInterface dialogInterface, int i)
                              {
75                            dialogInterface.dismiss();
76                            finish();
77                            Intent intent = new Intent(getApplicationContext(),
                                  LoginActivity.class);
78                            startActivity(intent);
79                        }
80                    });
81                }
82                AlertDialog alertDialog = builder.create();
83                alertDialog.show();
84            }
85        });
86
87        btnBack.setOnClickListener(new View.OnClickListener() {
88            @Override
89            public void onClick(View view) {
90                onBackPressed();
91            }
92        });
93    }
94 }
```

Listato 4.15: Classe per caricare tutti i dettagli di una ricetta

4.1.6 Autenticazione

La classe `AccessLoader`, mostrata nel Listato 4.16, gestisce il thread che ha il compito di connettersi con il database, inviando i dati per l'autenticazione dell'utente. Questa classe sarà chiamata da due classi differenti, illustrate successivamente in questa sezione. Infatti, `AccessLoader` invia i dati di autenticazione, ma questi variano a seconda che l'utente stia eseguendo il login o si stia registrando.

```
1  public class AccessLoader extends AsyncTask<String, Void, String> {
2
3      private Context context;
4      private Activity activity;
5      private AlertDialog.Builder builder;
6      private ProgressDialog progressDialog;
7      private CountDownLatch latch;
8      private static User user;
9
10
11     public AccessLoader(Context context, CountDownLatch latch) {
12         this.context = context;
13         this.activity = (Activity) context;
14         this.latch = latch;
15         builder = new AlertDialog.Builder(activity);
16     }
17
18     @Override
19     protected void onPreExecute() {
20         builder = new AlertDialog.Builder(activity);
21         progressDialog = new ProgressDialog(context);
22         progressDialog.setTitle("Please wait...");
23         progressDialog.setMessage("Connecting to server");
24         progressDialog.setIndeterminate(true);
25         progressDialog.setCancelable(false);
26         progressDialog.show();
27     }
28
29     @Override
30     protected String doInBackground(String... params) {
31
32         String method = params[0];
33         try {
34             if (method.equals("register")) {
35
36                 URL url = new URL(Constant.URL_REGISTER);
37                 String name = params[1];
38                 String lastName = params[2];
39                 String email = params[3];
40                 String password = params[4];
41                 String address = params[5];
42                 JSONObject jsonObject = new JSONObject();
43                 jsonObject.put("name", lastName);
44                 jsonObject.put("lastName", name);
45                 jsonObject.put("email", email);
46                 jsonObject.put("password", password);
47                 jsonObject.put("address", address);
48                 String json = connectionResult(url, jsonObject);
49                 if(onPost(json)){
50                     PreferenceUtils.saveEmail(email, context);
51                 }
52                 latch.countDown();
53                 return json;
54
55             } else if (method.equals("login")) {
56
57                 URL url = new URL(Constant.URL_LOGIN);
58                 String email = params[1];
59                 String password = params[2];
```

```
60              JSONObject jsonObject = new JSONObject();
61              jsonObject.put("email", email);
62              jsonObject.put("password", password);
63              String json = connectionResult(url, jsonObject);
64
65             progressDialog.dismiss();
66
67              if(onPost(json)){
68                  saveUser(json);
69              }
70              latch.countDown();
71              return json;
72          }
73      } catch (MalformedURLException | JSONException e) {
74          e.printStackTrace();
75      }
76      return null;
77  }
78
79  private String connectionResult(URL url, JSONObject jsonObject) {
80      try {
81          HttpURLConnection httpURLConnection = (HttpURLConnection) url.
                openConnection();
82          httpURLConnection.setRequestMethod("POST");
83          httpURLConnection.setRequestProperty("Content-Type", "application/json"
                );
84          httpURLConnection.setRequestProperty("Accept", "application/json");
85          httpURLConnection.setDoOutput(true);
86          httpURLConnection.setRequestProperty("Connection", "Keep-Alive");
87          httpURLConnection.setDoInput(true);
88          httpURLConnection.connect();
89          DataOutputStream outputStream = new DataOutputStream(httpURLConnection.
                getOutputStream());
90          outputStream.write(jsonObject.toString().getBytes(Constant.ENCODING));
91
92          int code = httpURLConnection.getResponseCode();
93          if (code == 200) {
94              InputStream inputStream = httpURLConnection.getInputStream();
95              BufferedReader bufferedReader = new BufferedReader(new
                    InputStreamReader(inputStream));
96              StringBuilder stringBuilder = new StringBuilder();
97              String line = "";
98              while ((line = bufferedReader.readLine()) != null) {
99                  stringBuilder.append(line);
100             }
101             Thread.sleep(2000);
102             httpURLConnection.disconnect();
103             String json = stringBuilder.toString().trim();
104             return json;
105         }
106     } catch (Exception e) {
107         e.printStackTrace();
108         AlertDialog.Builder builder = new AlertDialog.Builder (context);
109         builder.setTitle("Connection failed");
110         builder.setMessage("Please check your connection");
111         builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
112             @Override
113             public void onClick(DialogInterface dialogInterface, int i) {
114                 dialogInterface.dismiss();
115             }
116         });
117         AlertDialog alertDialog = builder.create();
118         alertDialog.show();
119         try {
120             Thread.sleep(3000);
121         } catch (InterruptedException ex) {
122             ex.printStackTrace();
123         }
124     }
125     return null;
```

```
126     }
127
128     private boolean onPost(String json) {
129
130         try {
131             JSONObject jsonObject = new JSONObject(json);
132             JSONArray jsonArray = jsonObject.getJSONArray("server_response");
133             JSONObject newJsonObject = jsonArray.getJSONObject(0);
134             String code = newJsonObject.getString("code");
135             //String message = newJsonObject.getString("message");
136
137             if (code.contains("true")){
138                 return true;
139             }
140             else {
141                 return false;
142             }
143
144         } catch (JSONException e) {
145             e.printStackTrace();
146         }
147         return false;
148     }
149
150     private void saveUser(String json) throws JSONException {
151         JSONObject jsonObject = new JSONObject(json);
152         JSONObject userJson = jsonObject.getJSONObject("user");
153         User.setName(userJson.getString("name"));
154         User.setLastName(userJson.getString("lastName"));
155         User.setLastName(userJson.getString("email"));
156         User.setPassword(userJson.getString("password"));
157         User.setAddress(userJson.getString("address"));
158         User.setIsLog(true);
159         PreferenceUtils.saveEmail(User.getEmail(), context);
160         PreferenceUtils.savePassword(User.getPassword(), context);
161     }
162
163     public static User getUser() {
164         return user;
165     }
166 }
```

Listato 4.16: Classe per inviare i dati inseriti dagli utenti e ricevere una risposta dal server

Login

La classe `LoginActivity` (Listato 4.17) inizializza i campi necessari per effettuare il login, ovvero richiede email e password dell'utente e prevede un pulsante per proseguire con l'accesso. I dati inseriti vengono inviati al database tramite la classe `AccessLoader` (Listato 4.16); infine, una risposta all'utente per notificare l'avvenuto accesso. Inoltre, nel caso in cui le credenziali per il login fossero errate o un campo venisse lasciato vuoto, viene inviato un messaggio che evidenzia l'errore.

```
1  public class LoginActivity extends AppCompatActivity {
2
3      private EditText email,password;
4      private Button loginButton;
5
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          super.onCreate(savedInstanceState);
9          setContentView(R.layout.login_form);
10         setTitle(R.string.app_name);
```

```
11
12          email = (EditText) findViewById(R.id.email_input);
13          password = (EditText) findViewById(R.id.password_input);
14          loginButton = (Button) findViewById(R.id.btn_login);
15
16          Object mail = PreferenceUtils.getEmail(LoginActivity.this);
17          if(mail != null){
18              email.setText((String) mail);
19          }
20
21          loginButton.setOnClickListener(new View.OnClickListener() {
22              @Override
23              public void onClick(View view) {
24                  if( email.getText().toString().equals("") || password.getText().
                        toString().equals("") ){
25                      AlertDialog.Builder builder = new AlertDialog.Builder(
                            LoginActivity.this);
26                      builder.setTitle("Something wrong...");
27                      builder.setMessage("Fields can't be empty!");
28                      builder.setPositiveButton("OK", new DialogInterface.
                            OnClickListener() {
29                          @Override
30                          public void onClick(DialogInterface dialogInterface, int i)
                                {
31                              dialogInterface.dismiss();
32                          }
33                      });
34                      AlertDialog alertDialog = builder.create();
35                      alertDialog.show();
36                  }
37                  else {
38                      CountDownLatch latch = new CountDownLatch(1);
39                      AccessLoader accessLoader = new AccessLoader((Context)
                            LoginActivity.this, latch);
40                      accessLoader.execute("login", email.getText().toString().trim()
                            , password.getText().toString().trim());
41                      try {
42                          latch.await();
43                      } catch (InterruptedException e) {
44                          e.printStackTrace();
45                      }
46                      if(User.getIsLog()){
47                          Intent intent = new Intent(LoginActivity.this,MainActivity.
                                class);
48                          startActivity(intent);
49                      }else{
50                          Toast.makeText(LoginActivity.this,"Login fallito, riprovare"
                                ,Toast.LENGTH_SHORT).show();
51                          email.setText("");
52                          password.setText("");
53                      }
54                  }
55              }
56          });
57
58          TextView registerTV = (TextView)findViewById(R.id.register_link);
59          registerTV.setMovementMethod(LinkMovementMethod.getInstance());
60          registerTV.setOnClickListener(new View.OnClickListener() {
61              @Override
62              public void onClick(View v) {
63                  Intent intent = new Intent(LoginActivity.this, RegisterActivity.
                        class);
64                  startActivity(intent);
65              }
66          });
67      }
68  }
```

Listato 4.17: Classe per la visualizzazione della schermata di Login

Registrazione

La classe `RegisterActivity` (Listato 4.18) ha lo stesso funzionamento della classe `LoginActivity` appena mostrata, con la differenza che non vengono inviati al database solo email e password, ma si richiede la compilazione di altri campi, ovvero nome, cognome, email e indirizzo. Per il resto è analoga alla precedente, infatti, anche questa classe invia un messaggio di avvenuta registrazione, o di errore, nel caso in cui l'utente non compila tutti i campi richiesti.

```
1  public class RegisterActivity extends AppCompatActivity {
2
3      private EditText name,lastName,email,password,password2,address;
4      private Button registerButton;
5
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          super.onCreate(savedInstanceState);
9          setContentView(R.layout.register_form);
10         setTitle(R.string.app_name);
11
12         name = (EditText) findViewById(R.id.name_input);
13         lastName = (EditText) findViewById(R.id.last_name_input);
14         email = (EditText) findViewById(R.id.email_input);
15         password = (EditText) findViewById(R.id.password_input);
16         password2 = (EditText) findViewById(R.id.password2_input);
17         address = (EditText) findViewById(R.id.address_input);
18         registerButton = (Button) findViewById(R.id.btn_register);
19         registerButton.setOnClickListener(new View.OnClickListener() {
20             @Override
21             public void onClick(View view) {
22                 if(name.getText().toString().equals("") || lastName.getText().
                       toString().equals("") || email.getText().toString().equals("")
                        || password.getText().toString().equals("") || password2.
                       getText().toString().equals("") || address.getText().toString
                       ().equals("")){
23                     AlertDialog.Builder builder = new AlertDialog.Builder(
                           RegisterActivity.this);
24                     builder.setTitle("Something wrong...");
25                     builder.setMessage("Fields can't be empty!");
26                     builder.setPositiveButton("OK", new DialogInterface.
                           OnClickListener() {
27                         @Override
28                         public void onClick(DialogInterface dialogInterface, int i)
                               {
29                             dialogInterface.dismiss();
30                         }
31                     });
32                     AlertDialog alertDialog = builder.create();
33                     alertDialog.show();
34                 }
35                 else if (!(password.getText().toString().equals(password2.getText()
                       .toString()))){
36                     AlertDialog.Builder builder = new AlertDialog.Builder(
                           RegisterActivity.this);
37                     builder.setTitle("Something wrong...");
38                     builder.setMessage("Passwords are not equal!");
39                     builder.setPositiveButton("OK", new DialogInterface.
                           OnClickListener() {
40                         @Override
41                         public void onClick(DialogInterface dialogInterface, int i)
                               {
42                             dialogInterface.dismiss();
43                             password.setText("");
44                             password2.setText("");
45                         }
46                     });
47                     AlertDialog alertDialog = builder.create();
```

```
48                  alertDialog.show();
49              }
50              else {
51                  CountDownLatch latch = new CountDownLatch(1);
52                  AccessLoader accessLoader = new AccessLoader((Context)
                        RegisterActivity.this,latch);
53                  accessLoader.execute("register", name.getText().toString().trim
                        (), lastName.getText().toString().trim(), email.getText().
                        toString().trim(), password.getText().toString().trim(),
                        address.getText().toString().trim());
54
55                  try {
56                     latch.await();
57                  } catch (InterruptedException e) {
58                     e.printStackTrace();
59                  }
60                  Intent intent = new Intent(RegisterActivity.this, LoginActivity
                        .class);
61                  startActivity(intent);
62              }
63          }
64      });
65
66      TextView loginTV = (TextView)findViewById(R.id.login_link);
67      loginTV.setMovementMethod(LinkMovementMethod.getInstance());
68      loginTV.setOnClickListener(new View.OnClickListener() {
69          @Override
70          public void onClick(View v) {
71              Intent intent = new Intent(RegisterActivity.this,LoginActivity.
                    class);
72              startActivity(intent);
73          }
74      });
75  }
76 }
```

Listato 4.18: Classe per la visualizzazione della schermata di Registrazione

4.1.7 Altre schermate

Esistono, anche, altre schermate dell'applicazione che sono state implementate, ma di cui non si riporta il codice in quanto non sono considerate essenziali ai fini del progetto. Si è deciso, in ogni caso, di spiegarne il funzionamento. Le schermate secondarie dell'app sono:

- *Schermata progetto*: questa activity consiste in una FAQ che l'utente può consultare per comprendere meglio il funzionamento dell'app e per scoprire qualcosa in più sui creatori del progetto.
- *Schermata profilo*: l'utente può accedere a questa activity soltanto se si è già autenticato. In tal caso, egli può modificare i propri dati, con la sola eccezione dell'email con cui si è registrato.
- *Schermata le tue ricette*: questa schermata contiene le ricette che l'utente ha deciso di salvare, con lo scopo di aiutare quest'ultimo a ritrovare una ricetta trovata in un momento precedente. Le ricette saranno visualizzate con la stessa modalità con cui vengono mostrate nella home page dell'applicazione, ma, ovviamente, la schermata in esame ne conterrà solo una porzione.

4.2 Manuale utente

Concludiamo questo capitolo sull'implementazione dell'applicazione EAToday con il manuale utente, che contiene gli screen delle schermate più importanti dell'app con la relativa spiegazione.

4.2.1 Schermata iniziale app

La home page di EAToday (Figura 4.1) coincide con la schermata che mostra le varie ricette. All'apertura dell'app, quindi, l'utente può iniziare a scorrere la home per scoprire le ricette disponibili, per poi premere su una di essa per visualizzarne i dettagli di preparazione. Su questa schermata vengono mostrate anche alcune caratteristiche importanti della ricetta, ovvero il tempo e la difficoltà di preparazione, il costo degli ingredienti e se è adatta a vegetariani e vegani, utili per la scelta della ricetta da preparare. In alternativa, l'utente può utilizzare la barra di ricerca, posta in basso al centro, per cercare una pietanza, scrivendo l'iniziale del nome della ricetta.

Inoltre, l'utente può premere sulle tre linee poste in alto a sinistra per accedere al menù e poter usufruire delle altre funzionalità dell'applicazione.

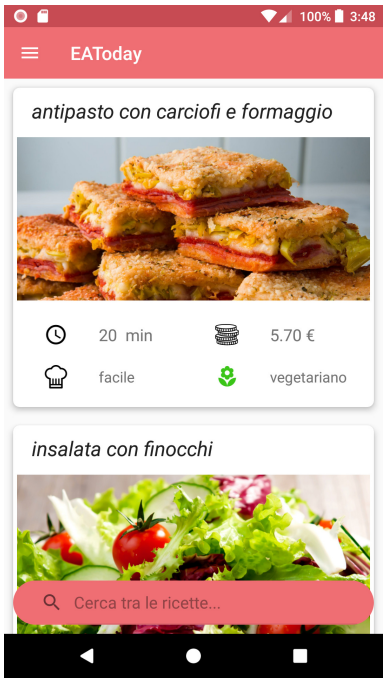


Figura 4.1: Screenshot della home page di EAToday

Drawer menù

In Figura 4.2 è mostrato il menù dell'applicazione. Come è stato scritto in precedenza, il menù è diverso a seconda che l'utente abbia effettuato il login o no. Quello in figura è il menù presente quando l'utente ha eseguito l'accesso. Si nota che le voci presenti sono "home", che permette di tornare alla schermata con tutte le ricette, "profilo", che permette di accedere ai propri dati personali per, eventualmente, modificarli, "filtra ricette", che consente di inserire dei requisiti che devono avere le ricette, così da restringere il campo per la ricerca, "progetto", ovvero la sezione che spiega lo scopo del progetto e chi sono i suoi ideatori, "logout", per disconnettersi dall'applicazione, e "le tue ricette", per visualizzare solo le ricette che l'utente ha, precedentemente, salvato. Nel caso in cui l'utente non avesse effettuato l'accesso, non ci sarà la schermata "profilo", bensì "login", che permette all'utente stesso di accedere o di registrarsi. Inoltre, non sarà presente né l'opzione "logout", né la sezione "le tue ricette", in quanto la possibilità di salvarsi le ricette è una prerogativa di chi è registrato ed ha effettuato l'accesso all'app.

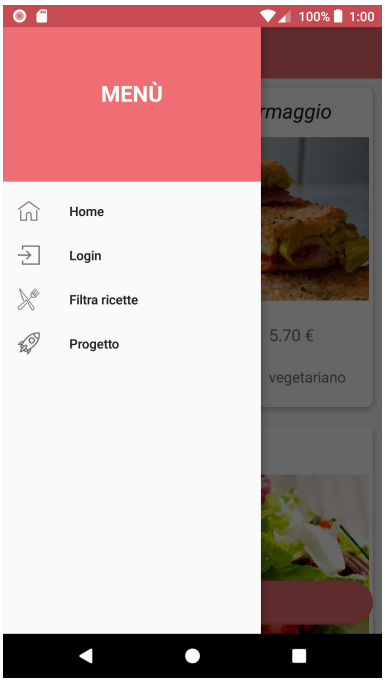


Figura 4.2: Screenshot del menù

Schermata relativa alla barra di ricerca

La barra di ricerca (Figura 4.3) è posizionata nella zona inferiore della home page. Il suo funzionamento è semplice ma utile per velocizzare l'utilizzo dell'applicazione; l'utente può, infatti, utilizzare tale barra per cercare una ricetta, inserendo le

lettere iniziali del nome della stessa. La schermata si aggiornerà immediatamente, mostrando le ricette il cui nome combacia con le lettere scritte.

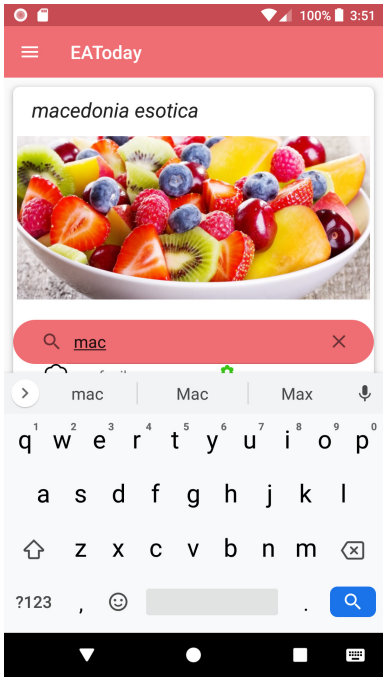


Figura 4.3: Screenshot della barra di ricerca

4.2.2 Schermata di filtro delle ricette

La schermata in Figura 4.4 mostra l'activity che viene aperta quando, dal menù, l'utente preme su "filtra ricette". Come si vede dalla foto, l'utente può stabilire degli ingredienti che devono essere inclusi e altri che devono essere esclusi. Per farlo basta scrivere il nome dell'ingrediente, o una parte del nome, e premere "aggiungi"; se ci sono più ingredienti che iniziano con le stesse lettere, l'applicazione proporrà una lista nell'ambito della quale selezionare l'ingrediente giusto. Nel caso in cui l'utente abbia inserito un ingrediente sbagliato, può eliminarlo semplicemente con uno swap, a destra o a sinistra. Inoltre, l'utente può scegliere se sta cercando un pasto per vegetariani o vegani, e questo si stabilisce mediante due switch. Per default, quando l'utente indica che vuole un pasto vegetariano, si attiva anche lo switch del pasto vegano, che, però, può poi essere tolto manualmente. Infine l'utente può selezionare il grado di difficoltà della ricetta desiderato, scegliendo tra "facile", "medio" e "difficile". Una volta completati tutti questi campi, l'utente può premere sul tasto "cerca" per visualizzare le ricette che combaciano con le sue richieste, oppure può premere su "indietro", per tornare alla home page.

Figura 4.4: Screenshot del filtro relativo alle ricette

4.2.3 Schermata relativa ai dettagli della ricetta

La schermata in Figura 4.5 è quella che si avvia quando l'utente, navigando per la home page, individua una ricetta che gli interessa e la seleziona per visualizzarne i dettagli. In questa activity vengono mostrati, innanzitutto, foto d'anteprima e nome delle ricetta. Subito sotto vengono indicati gli ingredienti necessari; se l'utente preme sul nome dell'ingrediente, apparirà una scritta che indica il prezzo dell'alimento per unità di vendita. Infine, è presente una descrizione, che indica la preparazione della ricetta scelta. Inoltre, in alto a destra, è presente il pulsante "salva", che permette di inserire la ricetta nella sezione "le tue ricette", in modo tale da poter essere trovata facilmente in un secondo momento.

4.2.4 Schermata login

In Figura 4.6 è mostrata la schermata che appare quando l'utente, non ancora connesso all'app, preme sulla voce "login" dal menù. L'applicazione richiede all'utente di inserire la propria email e la propria password, per poi premere sul tasto "login". Se le credenziali d'accesso sono corrette, egli verrà reindirizzato sulla home page; altrimenti, se le credenziali sono errate o è stato lasciato un campo vuoto, l'app invia un messaggio d'errore, ma si rimane sulla stessa activity, così da permettere un nuovo tentativo d'accesso. Infine, nella parte inferiore della schermata, è presente la scritta "Sei nuovo? Registrati qui.", così, se l'utente non si è mai registrato, può premere sulla scritta per essere reindirizzato alla pagina giusta.

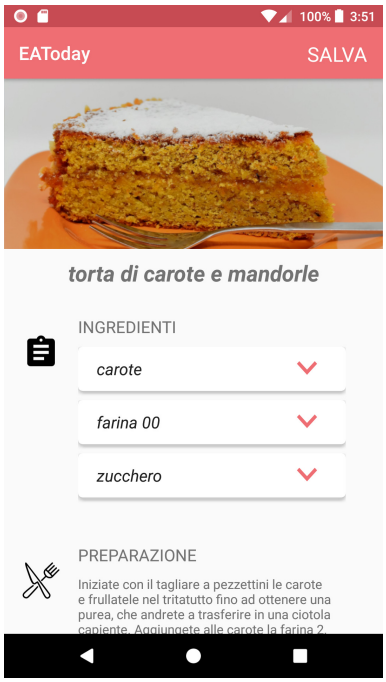


Figura 4.5: Screenshot dei dettagli di una ricetta

4.2.5 Schermata di registrazione

In Figura 4.7 è mostrata la schermata per la registrazione. L'applicazione richiede al nuovo utente di inserire il proprio nome, cognome, la propria email, la password che vuole utilizzare e il proprio indirizzo. Questi campi sono seguiti dal tasto "registrati", per procedere con la registrazione. Se questa va a buon fine, l'utente verrà reindirizzato alla home page, altrimenti comparirà un messaggio d'errore e l'utente potrà eseguire un nuovo tentativo. Nella parte inferiore è presente la scritta "Sei già registrato? Clicca qui.". In questo modo l'utente che si è già registrato può premere sulla scritta per accedere velocemente alla schermata per il login, spiegata precedentemente.

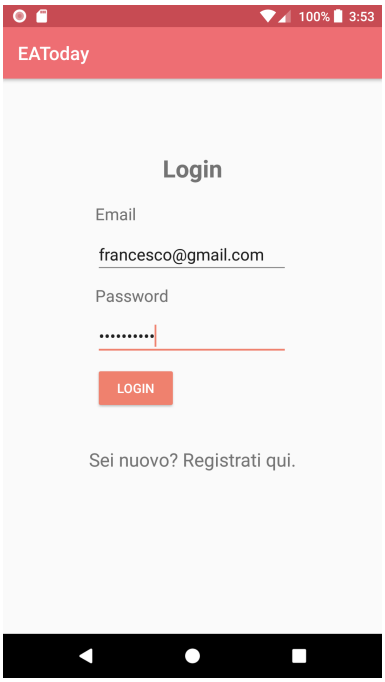


Figura 4.6: Screenshot della schermata login

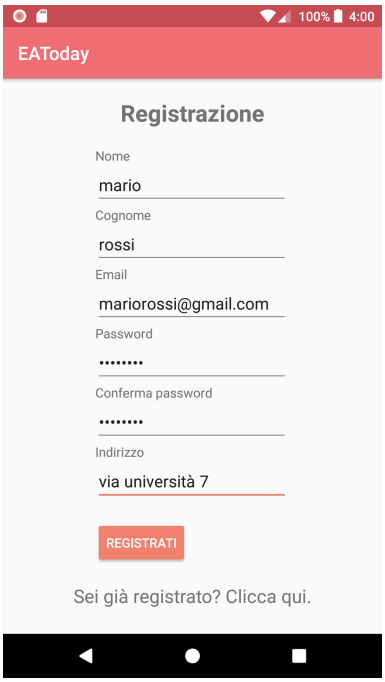


Figura 4.7: Screenshot della schermata registrazione

5

Discussione, conclusioni e sfide future

Questo capitolo inizierà con una discussione critica in merito al lavoro realizzato, cercando di evidenziare i punti di forza e di debolezza. Successivamente trarrà le conclusioni in merito al lavoro svolto e delineerà alcuni possibili sviluppi futuri.

5.1 Discussione

EAToday potrebbe rappresentare una svolta contro lo spreco alimentare. Attualmente, diversi player stanno sperimentando svariate tecniche per provare ad alleviare questo problema. L'applicazione risulterebbe utile a ridare una possibilità a tutti quei prodotti che, se non combinati in piatti per essere consumati nel breve periodo, verrebbero difficilmente acquistati. Il tema della "Circular Economy" è tra i più gettonati in questo periodo, e l'uscita di un progetto del genere calvacherebbe di sicuro quest'onda; bisognerà, però, tener conto del rapporto che hanno le persone con gli alimenti in scadenza, e far capire che questi prodotti sono ancora buoni.

Inoltre, l'applicazione permette di filtrare le ricette per ingredienti già in possesso, in modo da poter ridurre ulteriormente lo spreco, utilizzando i prodotti già presenti nelle case dei consumatori.

5.1.1 Punti di forza e di debolezza di EAToday

Di seguito verranno illustrati i punti di forza e di debolezza relativi all'intero progetto e quelli relativi all'applicazione fin qui implementata.

Punti di forza

- *Mercato e scalabilità*: il mercato attuale permette una rapida diffusione di idee simili, nonchè garantisce al progetto un'ottima oppurtunità dovuta alla sua scalabità, in quanto esso è facilmente replicabile in moltissime città medio-grandi informaticamente attrezzate. Inoltre, l'unico vero competitor, attualmente, è "Too Good To Go", che si basa sullo stesso principio di EAToday, ma non si occupa di consegna e combinazione in ricette.

- *Circular Economy e finanziamenti*: il concetto di ridare vita ai prodotti in scadenza sposa a pieno il concetto di "Circular Economy", e il progetto potrebbe essere beneficiario di diversi tipologie di concorsi e bandi, come, per esempio, quelli emessi dall'Unione Europea.
- *Design dell'app*: il Design dell'app risulta essere molto gradevole, realizzato in maniera minimale, così da rendere l'app intuitiva e di facile comprensione, in modo tale da poter raggiungere anche le persone meno esperte di applicazioni generiche.
- *Nome*: il nome fa la sua parte; è breve, piacevole ed esprime in modo chiaro il concetto dietro a tutto il progetto.

Punti di debolezza

- *Diffusione e target*: un progetto come EAToday, nonstante la sua scalabilità, perderebbe di efficacia nei centri abitati medio-piccoli. Oltretutto il target deve essere dinamico e di larghe vedute per apprezzare un progetto del genere. In sostanza, quindi, prima di portare EAToday in una determinata città, bisognerà studiarne bene caratteristiche e gli abitanti.
- *Gestione dei rivenditori*: per quanto l'app possa risultare semplice da utilizzare per gli utilizzatori finali, che la vedono esattamente come è stata fin ad ora implementata, tutto ciò che c'è dietro sarà molto più complesso, perchè bisognerà gestire i vari negozi aderenti ed istruirli a caricare i prodotti nel database.
- *Preparazione ricette*: la schermata che visualizza come preparare le ricette risulta essere troppo semplicistica allo stato attuale, per questo motivo andrà arrichita di particolari funzioni interattive. Inoltre, si potrebbe pensare di delegare il problema ad una figura terza, come, per esempio, una rivista online di cucina, per poi stringere una collaborazione e affidare ad essa la gestione della composizione delle ricette.

5.1.2 Lesson Learned

Durante la stesura di questa tesi ci siamo imbattuti in diversi aspetti che ci hanno aiutato a crescere in vista di progetti futuri.

La progettezione di un'app potrebbe richiedere moltissime rivisitazioni dovute a dettagli che sorgono solo in fase di implementazione. Quindi è molto importante avere piena comprensione di tutti i requisiti, nonchè progettare l'app con solide basi, che rendano possibile l'aggiunta di funzioni in tempi anche significativamente distanti tra loro, senza compromettere la struttura complessiva.

Android Studio permette di implementare svariati programmi, per diversi hardware, in moltissimi modi diversi. Tutto ciò rappresenta a pieno il concetto di "non si smette mai di imparare". In effetti è proprio vero, potremmo tranquillamente affermare di riuscire a rifare l'app da capo in molto meno tempo e in modo più efficace ed efficiente.

Di fondamentale importanza è stata, anche, l'organizzazione del codice, in quanto svilupparlo in maniera ordinata permette la manutenibilità dello stesso, così da rendere anche più sempice il *debugging*.

Un altro aspetto importante è la gestione del tempo; infatti, dividere l'intero percorso in mini-obiettivi e associare delle *deadline* in tempi stretti, consente di organizzarsi al meglio, evitando di lasciarsi tanto lavoro all'ultimo.

5.2 Conclussioni e sfide future

In questa tesi, siamo partiti dall'impatto che Android ha avuto nelle nostre vite, analizzandone le caratteristiche e gli sviluppi negli anni, per poi parlare nello specifico dell'*IDE Android Studio*, utile per l'implementazione di applicazioni.

In seguito, dopo una breve descrizione generale dell'app EAToday, abbiamo analizzato le ragioni che hanno portato allo sviluppo del progetto, descrivendone i requisiti da sodddisfare e le funzionalità da fornire agli utilizzatori finali.

A questo punto, abbiamo dapprima progettato l'app, sviluppandone il back-end e i wireframe per la realizzazione del front-end; successivamente, abbiamo implementato il tutto, con tanto di manuale utente per spiegare nello specifico le diverse funzionalità presenti nell'app.

Infine, abbiamo tratto alcune conclusioni, analizzando i punti di forza e di debolezza del progetto.

In conclusione, il percorso di EAToday è appena iniziato e lascia già molto spazio alle necessarie implementazioni future.

EAToday non è ancora un progetto pronto per un lancio sul mercato, ma presenta buone potenzialità, nonchè una solida struttura di partenza, definita in questa tesi.

5.2.1 Miglioramenti Futuri

Per il futuro bisognerà implementare funzioni più avanzate, che, però, hanno priorità massima per garantire la buona riuscita del progetto.

L'uso della localizzazione, che sia o meno gestita tramite i servizi di *Google Maps*, è un servizio importante che non è stato approfondito nell'implementazione dell'app. Le ricette e gli ingredienti erano chiaramente simulati, ma, quando poi avverà la *release* ufficiale, bisognerà fare in modo che nella combinazione delle ricette venga considerata la posizione del cliente, in modo tale da mostrare solo i prodotti entro un certo raggio.

Per quanto riguarda la questione della *cybersecurity*, l'app, attualmente, non ha un sistema di protezione dati. Non rispetta, infatti, le leggi sulla privacy e sul trattamento dei dati. Per rendere la sicurezza dell'app a norma, bisognerà mantenere la sessione utente tramite dei *token*, forniti da servizi terzi, e allinearsi con le varie leggi in materia. Inoltre, bisognerà proteggere l'intero sistema da vari tipi di attacchi, come, per esempio, l'*SQL injection*.

Riferimenti bibliografici

1. Smartphone Android nel mondo. `https://www.freedomapkapp.net/category/how-has-android-changed-the-world/`, 2017.
2. Emergenza spreco alimentare. `https://www.repubblica.it/cronaca/2019/02/04/news/giornata_spreco_alimentare-218246948/`, 2018.
3. Altervista. `https://it.wikipedia.org/wiki/AlterVista`, 2019.
4. Android. `https://it.wikipedia.org/wiki/Android`, 2019.
5. Android Studio. `https://it.wikipedia.org/wiki/Android_Studio`, 2019.
6. JSON. `https://www.json.org/json-it.html`, 2019.
7. Logo. `https://www.flaticon.com/free-icon/grocery_1411456`, 2019.
8. PhpMyAdmin. `https://it.wikipedia.org/wiki/PhpMyAdmin`, 2019.
9. M. Carli. *Android 6. Guida per lo sviluppatore*. Apogeo, 2016.
10. C. De Sio Cesari. *Manuale di Java 8: Programmazione orientata agli oggetti con Java standard edition 8*. Hoepli, 2014.
11. E. Cisotti and M. Giannino. *Android. Guida completa*. Edizioni LSWR, 2015.
12. F. Collini, M. Bonifazi, A. Martellucci, and S. Sanna. *Android: Programmazione avanzata*. Edizioni LSWR, 2015.
13. E. Frontoni and A. Mancini. *Programmazione ad oggetti per l'ingegneria informatica*. McGraw-Hill Education, 2019.
14. J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. *The Java Language Specification, Java SE 8 Edition (Java Series)*. Addison-Wesley Professional, 2014.
15. C. Horstmann. *Concetti di Informatica e fondamenti di Java (IV Edizione)*. Apogeo S.R.L., 2007.
16. G. Nudelman. *Android Design Patterns: Interaction Design Solutions for Developers*. Wiley, 2013.
17. H. Schildt. *Java. La guida completa*. McGraw-Hill Education, 2012.
18. B. Cruz Zapata. *Android Studio Application Development*. Packt Publishing, 2013.

Ringraziamenti

Penso che le esperienze maturate negli ultimi tre anni di università siano state molto formative. Sono consapevole di aver messo molto impegno nella mia carriera universitaria, ottenendo anche ottimi risultati in attività parallele.

Detto ciò, non sarei arrivato a questo punto se non fosse stato per diverse persone che ho il dovere di ringraziare.

Il più grande ringraziamento va ai miei genitori, che, con tanto affetto e molta pazienza, mi hanno da sempre fornito sostentamento economico e morale, facendo sempre il tifo per me.

Desidero ringraziare anche mio fratello, mia sorella e tutto il resto della mia famiglia per l'aiuto ed il sostegno che, nel corso di questi anni, anche da lontano, mi hanno dato.

Ma un ringraziamento particolare va ai miei nonni, che si sono sempre interessati ai miei progressi negli studi, credendo fermamente nelle mie potenzialità.

Desidero, poi, rivolgere un grande ringraziamento alla mia ragazza, Chiara, che mi ha sempre supportato e sopportato, dandomi la forza necessaria ad affrontare le varie difficoltà che mi si presentavano, sia nell'ambito universitario che nella vita di tutti i giorni.

Ringrazio, inoltre, gli amici di Loreto, di Ancona, il mio team di sviluppo per svariati progetti e i miei compagni di studi, con i quali ho sempre lavorato come una squadra. In modo particolare ringrazio Pietro che, oltre ad essere un ottimo coinquilino e compagno di squadra, ha partecipato con me a questo e tanti altri progetti, i quali hanno sempre portato ad ottimi risultati.

Infine, l'ultimo ringraziamento, ma non per importanza, va al mio relatore, il Prof. Domenico Ursino, che è stato molto presente e disponibile durante l'ideazione e la stesura di questa tesi. Lo ringrazio per avermi offerto la possibilità di prendere parte alla programmazione di un'applicazione Android, per aver sempre ascoltato le mie problematiche e i miei dubbi, fornendomi, ogni volta, ottime delucidazioni per arrivare alla fine di questo percorso.