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 Xamarin per suggerire all'utente ricette volte alla lotta allo spreco alimentare

Design and implementation of a Xamarin app for providing users with recipes conceived to reduce food waste

Relatore

Prof. Domenico Ursino

Candidato

Bogdan Petru Borc

Anno accademico 2018-2019

Non arrenderti mai, perché quando pensi che sia tutto finito, è il momento in cui tutto ha inizio.

Indice

Elenco delle figure

Elenco dei listati

Introduzione

Gli ultimi anni sono stati caratterizzati da un'ampia diffusione dei dispositivi *mobile*, tra cui, smartphone, tablet e, più recentemente, smartwatch. Al giorno d'oggi, il numero di persone che utilizza tali dispositivi cresce sempre di più. Infatti, solo in Italia, 8 persone su 10 posseggono uno smartphone e il numero di tali persone cresce sempre di più. Grazie ai servizi offerti dai dispositivi *mobile*, si sta verificando un abbandono progressivo dell'utilizzo del computer anche in ambito lavorativo e questo fa si che le grandi e piccole aziende debbano rendere disponibili i propri servizi anche per tali dispositivi. Tuttavia, a causa della frammentazione del mercato dei sistemi operativi mobile, le aziende si trovano spesso a dover decidere per quali piattaforme realizzare le proprie App.

Le App si dividono in 3 grandi categorie:

- *Native*: sono applicazioni sviluppate specificamente per un sistema operativo mobile; tali App riescono a sfruttare al massimo l'hardware del dispositivo offrendo prestazioni ottime a chi la utilizza.
- *Web App*: sono siti web "responsive", ovvero in grado di riconoscere il dispositivo utilizzato dall'utente e di addattarsi a quest'ultimo; non sono in grado di interagire con l'hardware e il software del dispositivo, quindi le prestazioni sono inferiori.
- *Ibride*: sono applicazioni che utilizzano la tecnologia web per la loro realizzazione riuscendo a sfruttare anche le risorse hardware del dispositivo.

La scelta di quale categoria di App realizzare non è affatto semplice. Infatti, tale scelta dipende da ciò che l'App deve fare. In particolare, se si vogliono avere prestazioni ottime, la maggior parte delle volte si sceglie quella nativa, ma questa scelta porta un grosso aumento dei prezzi a causa della necessità di sviluppare da zero l'applicazione per ogni piattaforma. Per questo motivo le aziende, soprattutto quelle più piccole, preferiscono realizzare le proprie App con un approccio ibrido. Rispetto alle App sviluppate in nativo, quelle ibride sono più rapide da sviluppare e meno dispendiose. Un altro importante vantaggio dello sviluppo ibrido sta nel fatto che viene generata una sola versione, indipendentemente dal numero di piattaforme sulle quali si desidera utilizzare l'applicazione. Tutti questi benefici comportano anche, dei punti a sfavore, uno dei quali è il fatto che le performance di tali applicazioni sono inferiori rispetto a quelle native.

Per la realizzazione delle App ibride ci sono molti framework tra cui Xamarin, che riesce a trovare un buon compromesso tra il metodo di sviluppo nativo e quello ibrido. Xamarin riesce, infatti, a garantire quasi la piena condivisibilità del codice fra le varie piattaforme e permette, anche, di avere ottime performance che si avvicinano molto a quelle native.

L'oggetto di tale tesi è, appunto, la realizzazione di un'applicazione mobile attraverso l'utilizzo del framework Xamarin. L'applicazione, chiamata EAToday, affronta un problema sempre più presente al giorno d'oggi, ovvero quello dello spreco alimentare. Lo "spreco alimentare" è il fenomeno che fa si che cibo ancora buono per essere consumato venga gettato. Si stima, infatti, che ogni anno un terzo di tutto il cibo prodotto per il consumo umano viene sprecato, soprattutto nei paesi ricchi.

Con l'App EAToday si prova a diminure il numero di quei prodotti che rischiano, ogni giorno, di essere inutilizzati e gettati nella spazzatura. Entrando nello specifico, l'applicazione deve mostrare all'utente diverse ricette realizzate utilizzando quegli alimenti che, nei vari negozi, sono a breve scadenza e rischiano di essere gettati. L'applicazione informa l'utente su come realizzare la ricetta voluta e di che ingredienti ha bisogno per la realizzazione. L'utente ha la possibilità di registrarsi e, una volta fatto ciò, può effettuare l'acquisto degli ingredienti di cui ha bisogno per la realizzazione della ricetta. L'utente ha, anche, la possibilità di filtrare le ricette per nome, per difficoltà, per tipo di ricetta, ma, soprattutto, per ingredienti: si ha, infatti, la possibilità di scegliere sia gli ingredienti voluti nella ricetta, sia quelli che, per vari motivi, si vuole evitare che siano presenti.

La tesi è strutturata come di seguito specificato:

- Nel Capitolo 1 si introduce il concetto di App in generale, si analizzano i vari tipi di App e vengono illustrate in modo più dettagliato le App ibride. Per quanto riguarda queste ultime, viene introdotto Xamarin, uno dei framework utilizzati per la loro realizzazione.
- Nel Capitolo 2 si ha una descrizione generale dell'App EAToday, analizzando i requisiti che l'applicazione dovrà rispettare e le funzionalità che essa dovrà offrire all'utente.
- Nel Capitolo 3 si progetta il database, soffermandosi sulla progettazione concettuale e su quella logica. Si analizzerà la struttura dell'App EAToday e si mostreranno i mockup dell'applicazione.
- Nel Capitolo 4 si analizzano gli strumenti utilizzati per lo sviluppo dell'applicazione; successivamente, si mostrano e si spiegano alcune parti di codice utilizzato per l'implementazione delle varie funzionalità dell'applicazione.
- Nel Capitolo 5 vengono analizzati i punti di forza e di debolezza dell'applicazione nonchè i possibili sviluppi futuri.

1

La programmazione ibrida e Xamarin

In questo capitolo si introduce il concetto di App in generale, si analizzano i vari tipi di App e vengono illustrate in modo più dettagliato le App ibride. Per quanto riguarda queste ultime, viene introdotto Xamarin, uno dei framework utilizzato per la loro realizzazione.

1.1 Classificazione generale delle App

Un'App è un software applicativo pensato per funzionare su vari dispositivi mobili, tra cui smartphone, tablet, smartwatch, etc. "App" è l'abbreviazione di applicazione mobile, che si differenzia da quelle tradizionali (per desktop) non solo per i dispositivi su cui viene utilizzata ma anche per il modo in cui viene realizzata; essa, infatti, è molto leggera, utilizzabile in maniera molto facile ma, soprattutto, è focalizzata su un servizio specifico. Come mostrato in Figura 1.1, le App si dividono in tre categorie, ovvero native, ibride e Web App.

Di seguito vengono introdotte queste tre categorie una per volta, cercando di comprendere le principali caratteristiche.

1.1.1 App native

Le App native sono applicazioni sviluppate specificamente per un sistema operativo mobile; quindi, per la loro realizzazione, si utilizza un linguaggio di programmazione diverso, in base alla piattaforma su cui si intende lavorare; ad esempio si utilizza Java per Android, Objective-C e Swift per iOS, C# e XAML per Windows Phone.

Tali App riescono a sfruttare al massimo l'hardware del dispositivo e, grazie all'utilizzo del Software Development Kit (SDK), interagiscono direttamente con le API della piattaforma, offrendo prestazioni migliori all'utente.

I principali vantaggi offerti delle App native sono:

- pieno accesso all'hardware del dispositivo: fotocamere, file system, rubrica etc.;
- interazione diretta con le API della piattaforma utilizzata, garantendo accesso immediato a tutte le funzionalità del dispositivo;

Figura 1.1: App nativa vs Web App vs App ibrida

- User Experience migliore grazie alla velocità, affidabilità, reactivity ma, soprattutto, alla risoluzione;
- notifiche che avvisano gli utenti;
- funzionamento offline.

Gli svantaggi sono dovuti principalmente alla complessità di sviluppo e alla forte dipendenza dello store nativo. L'App realizzata è utilizzabile solo sulla piattaforma per cui è stata pensata ed, in caso di aggiornamenti futuri, questi devono essere approvati prima della loro pubblicazione negli store. Le App native forniscono un'esperienza ideale all'utente grazie al fatto che riescono a sfruttare appieno le funzionalità del dispositivo; risultano, però, convenienti solo se si ha a disposizione un team di programmatori con ottime competenze nei linguaggi utilizzati dalle varie piattaforme.

1.1.2 Web App

Le Web App (Figura 1.2) sono quelle applicazioni che vengono archiviate su un server remoto e consegnate sul browser tramite un'interfaccia. Sono siti web "responsive", ovvero in grado di riconoscere il dispositivo utilizzato dall'utente e di adattarsi a quest'ultimo. Per il loro utilizzo si ha bisogno di una connessione Internet ma non è richiesta alcuna installazione; ciò comporta un minor utilizzo della memoria del dispositivo, anche se si hanno tempi di attesa più lunghi dovuti sia alla risposta del server, sia alla connessione Internet. Le Web App non sono in grado di interagire con l'hardware e il software del dispositivo.

I principali vantaggi delle Web App sono:

- nessuna richiesta di installazione per il loro utilizzo;
- capacità di adattarsi e di funzionare su tutte le piattaforme e tutti i dispositivi, indipendentemente dalla dimensione;
- sviluppo semplice grazie alla conoscenza, da parte di molti sviluppatori, dei linguaggi di programmazione utilizzati (HTML e CSS);
- tempi di sviluppo brevi;

Figura 1.2: Web App

- mancanza di necessità di approvazione, da parte degli store, per la loro pubblicazione.

Gli svantaggi sono:

- necessità di connessione Internet per accedere al browser e far funzionare la Web App;
- tempo di accesso più lungo a causa della mancata icona sul dispositivo;
- difficoltà di interagire con l'hardware dei dispositivi.

1.1.3 App ibride

Le App ibride sono quelle che riescono a mettere insieme i vantaggi delle App native e quelli delle Web App; in particolare, esse riescono a utilizzare le tecnologie web per sviluppare l'applicazione senza l'obbligo di una connessione Internet, garantendo, comunque, la possibilità di sfruttare le risorse locali del dispositivo.

Rispetto alle App native, esse sono più rapide e meno care da sviluppare, ma hanno lo svantaggio di fornire performance inferiori. Esiste la possibilità di avvicinarsi al funzionamento di un App nativa, ma questo vuol dire costi di sviluppo molto elevati, senza il pieno raggiungimento dell'obiettivo. Una caratteristica molto importante di questa tipologia di App è il fatto che viene generata una sola versione, indipendentemente dal numero di piattaforme su cui essa sarà utilizzata; questo è molto utile in caso di un aggiornamento dell'applicazione, in quanto non si ha la necessità di effettuare i vari cambiamenti su più piattaforme, ma è sufficiente apportare modifiche una sola volta.

Un'App ibrida (Figura 1.3) contiene un'istanza di browser (detta WebView) chiusa all'interno di un involucro di app nativa che è in grado di interagire con la piattaforma del dispositivo (quindi si ha la possibilità di accedere all'hardware di quest'ultimo) e con la WebView stessa. Ciò permette all'App ibrida di contenere tutte le risorse necessarie affinché il suo funzionamento si avvicini il più possibile a quello delle App native. Le performance di un'App ibrida non possono essere paragonate a quelle di un'App nativa; infatti quest'ultima non ha bisogno di alcun interprete JavaScript o motore di rendering HTML/CSS per il funzionamento al contrario, invece, di quanto avviene per le App ibride, dove questi fattori aggiungono un costo computazionale che non può essere trascurato.

Figura 1.3: App ibrida

Vantaggi

Di seguito sono indicati i vantaggi che l'approccio ibrido fornisce all'utilizzatore:

- *Indipendenza dallo store*: gli sviluppatori hanno la possibilità di eseguire operazioni di update/rollback dei contenuti o dell'applicazione stessa, senza richiedere agli utenti di aggiornare quest'ultima per mezzo dell'app store nativo.
- *Cross-platform*: gli sviluppatori costruiscono l'app una sola volta e la distribuiscono a piattaforme differenti.
- *Tempo di sviluppo breve*: costruire un'applicazione ibrida è tipicamente più veloce e richiede conoscenze standard altamente riutilizzabili.
- *Contenimento dei costi*: si ha un'unica applicazione per più piattaforme il che consente di risparmiare notevolmente sui costi.
- *Tanti sviluppatori*: ci sono tanti esperti con le conoscenze necessarie per lo sviluppo di tali app grazie al fatto che essa utilizza, per molti aspetti, la tecnologia Web.
- *Distribuzione e approvazione delle app*: proprio come le App native, quelle ibride vengono distribuite negli store di appartenenza. La loro approvazione richiede una sola sottomissione allo store; approvata la richiesta iniziale, si ha la possibilità di apportare aggiornamenti successivi attraverso il server, nel qual caso si ha la necessità di una nuova richiesta di sottomissione nel momento in cui uno sviluppatore intende modificare il codice nativo dell'App.
- *Accesso off-line*: le app ibride, come quelle native, possono essere utilizzate anche senza la connessione Internet.

Svantaggi

Le App ibride presentano anche alcuni svantaggi, ovvero:

- *Basse performance*: rispetto alle App native, la performance delle app ibride è inferiore. In particolare, esse possono essere più lente per quanto riguarda la riproduzione delle animazioni, i tempi di reazione e la precisione dell'utilizzo tramite touch screen.
- Le App ibride non possono essere aperte direttamente dal browser.

- *Interfaccia utente inconsistente*: potrebbe essere difficile simulare il look and feel delle interfacce utente delle specifiche piattaforme usando HTML, CSS e JavaScript.
- *Nessun controllo nativo dell'interfaccia utente*: senza un opportuno tool di supporto gli sviluppatori dovrebbero realizzare tutti gli elementi dell'interfaccia utente.
- *Limitazioni WebView*: l'applicazione funziona tramite l'istanza WebView; per questo le performance sono strettamente legate alle qualità del browser della piattaforma.
- *Accesso alle caratteristiche del dispositivo tramite plugin*: l'accesso alle API native del dispositivo necessita di plugin sviluppati in parallelo che lo supportino.

1.2 Xamarin

1.2.1 Introduzione

Xamarin è un framework utilizzato per lo sviluppo di App native e cross-platform. Un framework è un'architettura logica di supporto su cui un software può essere progettato e realizzato, con lo scopo di facilitare lo sviluppo da parte del programmatore. Il framework si basa sul progetto open source Mono, offrendo pieno supporto alle piattaforme Android, iOS e Windows Phone. Xamarin nasce nel 2011 negli USA da una software house fondato da Friedman e Icaza. I due, essendo sviluppatori del mondo Microsoft, basarono il loro software su C#.

L'obiettivo di Xamarin è quello di offrire agli sviluppatori uno strumento semplice e veloce per programmare applicazioni sulle diverse piattaforme basandosi su un unico linguaggio e provando a garantire prestazioni vicine a quelle native. Nel 2016, grazie all'acquisto di Xamarin da parte della Microsoft, iniziò una nuova era per lo sviluppo di App, grazie al fatto che il framework diventò open source e tantissimi sviluppatori indipendenti e piccoli team iniziarono ad utilizzarlo.

1.2.2 Strumenti offerti

Xamarin è costituito da tre componenti principali:

- `Xamarin.Android`: permette lo sviluppo di app Android tramite un wrapping delle API native in C#.
- `Xamarin.iOS`: permette lo sviluppo di app iOS tramite un wrapping delle API native in C#.
- `Xamarin.Forms`: si pone al di sopra delle due precedenti permettendo lo sviluppo di un'app completamente cross-platform.

Grazie a questi componenti è possibile gestire in C# tutte le caratteristiche delle varie piattaforme, dall'interfaccia utente alle risorse hardware del dispositivo.

In Figura 1.4 vengono mostrati i componenti di Xamarin e il loro legame con le SDK native.

Tale suddivisione permette di gestire l'interfaccia utente grazie a:

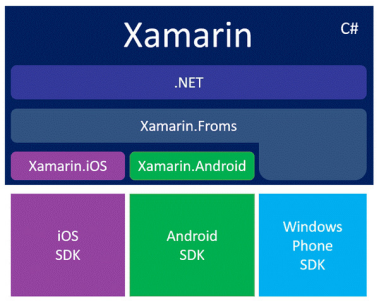


Figura 1.4: Struttura del framework Xamarin

- elementi tipici delle singole piattaforme (grazie al bilding con le SDK native);
- elementi definiti in `Xamarin.Forms` , nel caso di app cross-platform, in cui è necessario mostrare agli utenti un'interfaccia più famigliare.

Di seguito si analizzano le componenti `Xamarin.Forms` e quelle native (`Xamarin.Android` e `Xamarin.iOS`) specificando quando è opportuno utilizzarle.

Xamarin.Forms

È una gamma di strumenti utilizzati per la creazione di applicazioni, le quali vengono realizzate una sola volta e sono indirizzate a più piattaforme, riuscendo a ridurre in maniera significativa i tempi di sviluppo. Tale riduzione è dovuta al fatto che circa l'80-95% del codice è multipiattaforma, mentre il restante 5-20% contiene i comandi che stabiliscono le connessioni con API native. Grazie a ciò anche l'aggiornamento e la manutenzione diventano più semplici, perché si deve modificare un solo codice per tutte le piattaforme.

`Xamarin.Forms` è basata su un'architettura a tre stati come si nota in Figura 1.5 :

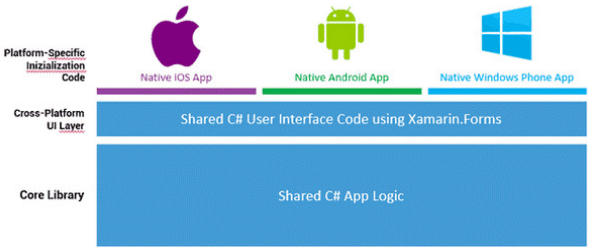


Figura 1.5: Struttura di Xamarin.Forms

I tre strati di cui si compone Xamarin.forms sono i seguenti:

- *Core Library*: contiene la logica dell'applicazione e di accesso ai dati definita in C#;
- *Cross-Platform UI Layer*: è lo strato contenente tutta la UI realizzata, definita in C#;
- *Platform-Specific Inizialization Code*: ha una piccola porzione di codice in C# nei progetti di ogni piattaforma ed è necessario per inizializzare ed eseguire l'app su ogni OS nativo.

`Xamarin.Forms` risulta particolarmente adatto per realizzare:

- applicazioni che richiedono funzionalità specifiche della piattaforma;
- applicazioni che richiedono inserimento di dati;
- applicazioni in cui la condivisione del codice è più importante dell'interfaccia utente personalizzata.

Xamarin.Android e Xamarin.iOS

Nonostante il fatto che l'approccio nativo di Xamarin è più dispendioso in termini di risorse e di tempo si preferisce utilizzarlo quando:

- L'applicazione presenta un elevato numero di elementi interattivi e di contenuti che si aggiornano dinamicamente; per tale App si vuole avere una grande efficienza in termini di prestazione.
- L'applicazione richiede un'interfaccia utente complessa: Android e iOS presentano grandi differenze dal punto di vista del design, per ciò che concerne i pulsanti, le barre, le notifiche istantanee, etc. Se l'App utilizza tali strumenti realizzarla con `Xamarin.Forms` sarebbe abbastanza difficile.
- L'applicazione deve occupare poco spazio in memoria: più le applicazioni sono piccole e più velocemente si caricano; infatti le applicazioni sviluppate con l'approccio nativo di Xamarin occupano meno memoria.

1.2.3 Alcuni concetti fondamentali: Activity, Layout e Views

Lo scopo principale di `Xamarin.Forms` è quello di unificare non solo la logica dell'app tra le varie piattaforme mobili, ma anche l'interfaccia utente. Per fare ciò, Xamarin fornisce un insieme di elementi e contenitori che compongono la grafica delle App, come mostrato in Figura 1.6.

Si può dividere l'interfaccia utente in tre grandi classi:

- *Page*: esiste una page per ogni schermata dell'App; Page viene detta "Activity" in Android e "ViewController" in iOS.
- *Layout*: è un "contenitore" che permette di definire e gestire al proprio interno altri layout e oggetti di tipo View.
- *View*: con questa classe ci si riferisce a oggetti dell'interfaccia, come pulsanti, etichette e riquadri per l'inserimento del testo.

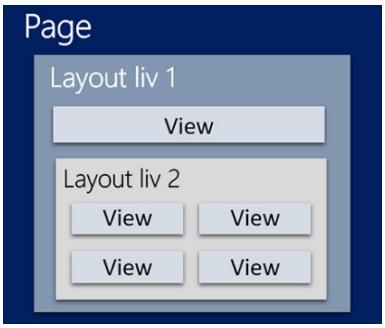


Figura 1.6: Elementi e contenitori da utilizzare nell'interfaccia utente di Xamarin

Activity

Le activity sono le classi chiave di Android in cui si svolgono tutte le azioni. L'activity di partenza di un'applicazione deve essere dichiarata dal programmatore. Ogni activity ha vari metodi:

- `onCreate()`: è chiamato quando l'activity viene creata per la prima volta. In questo metodo si possono inserire gestori di pulsanti, stili, etc.È sempre seguito da `onStart()`.
- `onStart()`: è chiamato quando l'activity sta diventando visibile all'utente; è seguito da `onResume()` se essa è in primo piano, altrimenti è seguito da `onStop()`.
- `onResume()`: è chiamato quando l'activity inizia ad interagire con l'utente. È sempre seguito da `onPause()`.
- `onPause()`: è chiamato quando l'activity perde lo stato di primo piano. Essa è ancora visibile all'utente; quindi è consigliabile mantenerla visivamente attiva e continuare ad aggiornare l'interfaccia utente. Le implementazioni di questo metodo devono essere molto rapide perché l'activity successiva non verrà ripresa fino a quando esso non verrà restituito. È seguito da `onResume()` se l'activity torna in primo piano o da `onStop()` se diventa invisibile per l'utente.
- `onStop()`: è chiamato quando l'activity non è più visibile per l'utente. Questo comando è in genere utilizzato per interrompere le animazioni e l'aggiornamento dell'interfaccia utente. È seguito da `onRestart()` se essa sta tornando per interagire con l'utente, o da `onDestroy()` se essa sta scomparendo.
- `onDestroy()`: rappresenta l'ultima chiamata che che viene ricevuta prima che l'activity venga distrutta. Ciò può accadere sia perché l'activity sta finendo sia perché il sistema sta temporaneamente distruggendo questa istanza dell'activity per risparmiare spazio.

Layout

Un layout è un "contenitore" che permette di definire e gestire al proprio interno altri layout e oggetti di tipo View, che consentono all'utente di interagire con l'applicazione. Grazie al layout è possibile definire la posizione e la dimensione dei vari

elementi contenuti al suo interno e organizzare opportunatamente i contenuti della pagina.

Le tipologie di layout supportate da `Xamarin.Forms` sono le seguenti (Figura 1.7):

- *AbsoluteLayout*: imposta la posizione e le dimensioni degli elementi figli utilizzando dei rettangoli o definendo delle specifiche proporzioni.
- *GridLayout*: crea un contenitore organizzato secondo una griglia con un certo numero di righe, colonne e con una larghezza opportuna.
- *RelativeLayout*: permette di definire dei vincoli tra gli elementi figli nonchè la posizione e la dimensione di questi ultimi al suo interno; questo è il layout che offre una maggiore libertà nella disposizione degli elementi e una maggiore fluidità.
- *StackLayout*: gli elementi figli contenuti in questo layout possono essere disposti verticalmente o orizzontalmente.
- *FlexLayout*: espone in modo efficiente i propri elementi garantendo l'ordine.

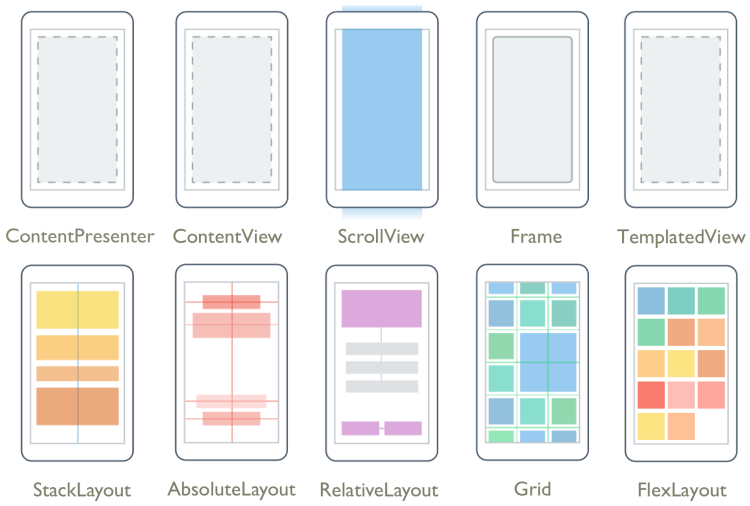


Figura 1.7: Differenza tra i vari layout

Oltre a questi layout, per rispondere a determinate esigenze, ne sono stati aggiunti altri più complessi. I più importanti sono:

- *ContentView*: definisce una schermata di base contenente un content al cui interno è presente una stringa di testo (label);
- *ScrollView*: è un contenitore generico a scorrimento sia verticale che orizzontale;
- *TemplatedView*:è una vista che visualizza il contenuto mediante un modello di controllo e la classe base per ContentView;
- *Frame*: un elemento che contiene un singolo elemento figlio, con alcune opzioni di framing.
- *ContentPresenter*: visualizza il contenuto di un *ContentControl*.

View

La classe `View` permette di definire diversi elementi grafici che compongono l'interfaccia utente, tra cui i layout e i controlli. Grazie a questi elementi l'utente può interagire con l'applicazione. Esistono vari tipi di View:

- *ActivityIndicator*: è un elemento animato che indica che un'operazione è in corso;
- *Button*: è un pulsante con cui interagire;
- *DatePicker*: permette di selezionare la data;
- *Editor*: è un campo modificabile per la scrittura di una o più linee di testo;
- *Entry*: è un campo modificabile per inserire un contenuto alfanumerico;
- *Image*: è una View che gestisce un'immagine;
- *Label*: è una stringa (a linea singola o multilinea) da visualizzare;
- *ProgressBar*: indica l'avanzamento di un'operazione;
- *Stepper*: permette di incrementare o decrementare un valore in un range definito.

2

Analisi dei requisiti

Questo capitolo inizia con una descrizione generale dell'App EAToday, analizzando, poi, i requisiti che l'applicazione dovrà rispettare e le funzionalità che essa dovrà offrire all'utente.

2.1 Descrizione generale App EAToday

EAToday è un'applicazione che è stata pensata per tentare di combattere lo spreco alimentare. Con questa espressione si intende lo sperpero di tutti quei prodotti che, nei vari negozi, come supermercati, fruttivendoli, macellerie, panifici, etc., hanno una scadenza a breve termine e che, per questo motivo, quando tale data di scadenza si avvicina, rischiano di essere gettati. Come si può osservare nell'articolo, riportato nella Figura 2.1, circa un terzo del cibo prodotto sul nostro pianeta viene sprecato ancora prima di arrivare a tavola; per questo motivo molti paesi hanno iniziato a definire alcune norme che vanno contro questo spreco. Carlo Petrini, fondatore del movimento internazionale *Slow Food*, fa notare che viene prodotto cibo per dodici miliardi di persone, quando, in realtà, la popolazione mondiale non arriva a otto miliardi. Questo fatto ci fa pensare a quanto cibo viene sprecato inutilmente. Si calcola che solo in Italia vengono buttate tra i rifiuti più di 4000 tonnellate di alimenti, arrivando a un totale europeo di 50000 tonnellate. Secondo le stime del *WWF*, globalmente si spreca una quantità di cibo che è pari a circa quattro volte la quantità di alimenti necessari a sfamare 800 milioni di persone denutrite. Questi dati sono estremamente allarmanti, ed è importante fare qualcosa per tentare di migliorare questa situazione, anche agendo nel piccolo del quotidiano.

Quest'App si pone nell'ottica di dare un piccolo contributo nel combattere lo spreco alimentare. Lo scopo è, infatti, quello di permettere all'utente di acquistare prodotti alimentari (Figura 2.2) con data molto vicina a quella della scadenza. L'App mostra delle ricette, che saranno selezionate da algoritmi che tengono conto degli alimenti in scadenza, disponibili nei vari negozi, e informa l'utente della difficoltà e del tempo di preparazione della ricetta selezionata. Di quest'ultima, l'utente può scegliere solo i prodotti di cui ha bisogno per la realizzazione della pietanza.

10 la Repubblica Lunedì 7 gennaio 2019

L'inchiesta

Alimenti, nuove date di scadenza per non buttare 500 miliardi di cibo

A&F

L'economia circolare

Primo piano

VITO DE CEGLIA, MILANO

Alleanza tra i giganti mondiali della grande distribuzione per creare un nuovo standard sui termini di consumo di frutta e ortaggi che sono troppo ravvicinati e moltiplicano gli scarti

Semplificare le etichette alimentari per ridurre il problema degli sprechi. Inizia a prendere forma la proposta, lanciata poco più di un anno fa, da Consumer Goods Forum (Cgf): organismo che riunisce 400 aziende del mondo retail di 72 Paesi tra cui distributori, produttori e fornitori con un giro di affari complessivo di 3,5 trilioni di euro e 100 milioni di lavoratori (diretti e indiretti). Comincia a concretizzarsi soprattutto negli Usa, Regno Unito e Francia dove alcune grandi aziende hanno accettato la sfida per migliorare la data di scadenza degli alimenti contrassegnati con la dicitura "best before", in italiano da "consumarsi preferibilmente entro" o "termine minimo di scadenza".

Tra queste vi sono nomi noti come Wal-Mart, Stores, Kellogg, Nestlé, Unilever, Bimbo, Carrefour e Tesco. Per l'Italia: Conad, Esselunga, Federdistribuzione, Indicod-Ecr Gs1 Italy, Illy, Barilla, Ferrero e Sofidel.

LA DISCRIMINANTE

La discriminante, secondo Cdf, è che riportare sull'etichetta una differente modalità di scadenza - da "consumarsi entro" (used by) o "preferibilmente" (best before) - può generare confusione o spingere il consumatore a gettare nella pattumiera cibo ancora buono da mangiare. Le stime parlano di 1,3 miliardi di tonnellate, per un valore di 1.000 miliardi di dollari all'anno (Fonte: Fao). Il gruppo di Cdf ha approvato, infatti, una "call action" per standardizzare le etichette alimentari entro il 2020.

E vuole farlo di concerto con "Champions 12.3": una coalizione di governi, imprese, organizzazioni internazionali, istituti di ricerca e società civile costituita con l'obiettivo di perseguire il diktat delle Nazioni Unite (SDG 12.3): ovvero, dimezzare lo spreco alimentare globale pro-capite a livello di vendita al dettaglio e di consumatori. In più, ridurre le perdite di cibo durante le catene di produzione e di fornitura, comprese le fasi del post-raccolto. Ma può davvero una dicitura diversa della data di scadenza arginare lo spreco alimentare?

IL TEST DI TESCO

La risposta arriva da Tesco, il primo rivenditore a rimuovere l'etichetta "best before", anche perché è il suo ceo, Dave Lewis, che guida l'alleanza di Champions 12.3. Il gruppo inglese a maggio l'ha tolta da 70 referenze di prima gamma (ortofrutta che non ha subito lavorazioni), per poi eliminarla un paio di mesi fa anche su altre 116 referenze. "La rimozione delle date entro cui consumare preferibilmente un alimento è il nostro modo per rendere più facile per i clienti ridurre gli sprechi alimentari a casa, e nel frattempo risparmiare", ha dichiarato di recente al "The Guardian" Mark Little, responsabile della riduzione dei rifiuti alimentari di Tesco. Per inciso: la decisione è stata approvata dalla quasi totalità dei clienti.

Anche l'Unione europea si è posta l'interrogativo sulle Tmc, pubblicando un anno fa un interessante studio: il quale conferma che "molti consumatori non capiscono le date riportate sull'etichetta dei prodotti alimentari, compresa la distinzione tra Tmc e scadenza obbligatoria". Tuttavia, lo studio osserva che "le insegne sono combattute se cambiare o meno la terminologia". Il problema evidentemente tocca anche l'Italia: un paese più virtuoso di altri - è il quarto al mondo nella lotta allo spreco alimentare -, ma dove c'è ancora molto da lavorare. A partire dai consumatori, l'anello debole della filiera considerato che al netto di 12 miliardi di euro di cibo sperperato, un quinto - cioè 3,5 miliardi - arriva da agricoltura (946 milioni), industria (1,1 miliardi) e distribuzione (1,4 miliardi). Il resto - 8,5 miliardi - è spreco domestico (Fonte: Osservatorio Waste Watcher).

IL CASO ITALIANO

E' quindi dentro le nostre case che dobbiamo intervenire. Perché sul fronte della distribuzione un contributo positivo l'ha dato la legge Gadda del 2016 che ha previsto semplificazioni burocratiche e sgravi fiscali a favore di chi dona cibo per fini di solidarietà. Sul fronte dei consumatori, invece, il nodo resta. E non so-

NEL MONDO

Lo spreco alimentare

Basterebbe 1/4 degli 1,30 miliardi di tonnellate di alimenti scaduti per sfamare 815 milioni di persone che non hanno accesso al cibo

Con i trend attuali stiamo consumando le risorse di 1 pianeta e mezzo

Che cosa buttiamo via ogni anno

CEREALI 30% · FRUTTA E VERDURA 45% · SEMI E LEGUMI 20%

LATTE 20% · PESCE 30% · CARNE 20%

IN ITALIA

L'Italia è uno dei paesi più virtuosi al mondo

Food Sustainability Index, Indice Internazionale di Sostenibilità Alimentare

sprechiamo solo il 2,3%

Tuttavia buttiamo via ancora troppo cibo

145 kg pro capite all'anno, l'equivalente di:

1.000 mele · 1.500 piatti di pasta · 750 baguette di legumi

12 miliardi € l'ammontare dello spreco alimentare

Famiglie (spreco domestico) 8.500 milioni €

Distribuzione (GDO) 1.444 milioni €

Produzione industriale 1.111 milioni €

Agricoltura 946 milioni €

FONTI: OSSERVATORIO WASTE WATCHER, FAO, BCFN

Dave Lewis ceo di Tesco e presidente di Champions 12.3

Marco Pedroni presidente Coop Italia

Giorgio Santambrogio ad gruppo Vegè

L'opinione

Su molti prodotti freschi interi i consumatori possono riconoscere da soli lo stato di buona conservazione

MARCO PEDRONI PRESIDENTE COOP ITALIA

L'opinione

Il valore di ciò che si sperpera già solo nel gap temporale tra 'best before' e 'use by', sono numeri da brivido

GIORGIO SANTAMBROGIO, AD GRUPPO VEGÈ

1,3 MILIARDI Sono le tonnellate di cibo che finisce tra i rifiuti per aver superato la data di scadenza

1000 MILIARDI È il valore espresso in dollari del cibo che ogni anno finisce tra gli scarti secondo la Fao

12 MILIARDI Tanto vale in euro lo spreco di cibo in Italia, 8,5 miliardi dei quali sono scarti domestici

Figura 2.1: Articolo che spiega lo spreco del cibo nel mondo

Tale applicazione ha molti benefici. I principali sono i seguenti:

- viene ridotto lo spreco di cibo;
- i prodotti a breve scadenza hanno un prezzo più basso (a volte addirittura del 50-75%) e questo aiuterà l'utente a risparmiare;

- viene supportato l'utente nella scelta di ricette diverse in base ai prodotti già disponibili, grazie ai filtri messi a disposizione nell'applicazione;
- viene fornito all'utente delle idee su ciò che vorebbe cucinare, presentandogli diverse ricette che può realizzare.

Figura 2.2: Principali prodotti a breve scadenza

2.2 Requisiti

Un requisito è una caratteristica che l'applicazione deve soddisfare. I requisiti si dividono in due categorie:

- *funzionali*: sono quelli che descrivono i servizi, o funzioni, offerti dall'applicazione;
- *non funzionali*: sono quelli che rappresentano vincoli sui servizi offerti dall'applicazione.

2.2.1 Requisiti funzionali

I requisiti funzionali che l'App "EAToday" dovrà garantire ai suoi utenti sono i seguenti:

1. permettere la visualizzazione di ricette;
2. permettere di filtrare le ricette in base agli ingredienti desiderati dall'utente;
3. permettere all'utente, se registrato, di ordinare gli ingredienti per una ricetta;
4. dare la possibilità di registrazione ai servizi offerti dell'app;
5. consentire il login ad un utente già registrato;
6. consentire all'utente di modificare il proprio profilo;
7. informare il cliente sulla difficoltà e il tempo necessario per la realizzazione di una ricetta.

2.2.2 Requisiti non funzionali

I requisiti non funzionali sono i seguenti:

1. l'applicazione dovrà essere realizzata attraverso la piattaforma Xamarin.
2. i linguaggi di programmazione utilizzati per la realizzazione dell'applicazione dovranno essere C# e XAML.

2.3 Diagramma dei casi d'uso

Il diagramma dei casi d'uso (Use Case Diagram) è una rappresentazione delle funzioni o servizi offerti da un sistema, così come essi sono percepiti e utilizzati dagli utenti che interagiscono col sistema stesso. Tale diagramma può essere considerato come uno strumento di rappresentazione dei requisiti funzionali di un sistema.

Nel caso dell'App "EAToday", il diagramma dei casi d'uso viene mostrato in Figura 2.3.

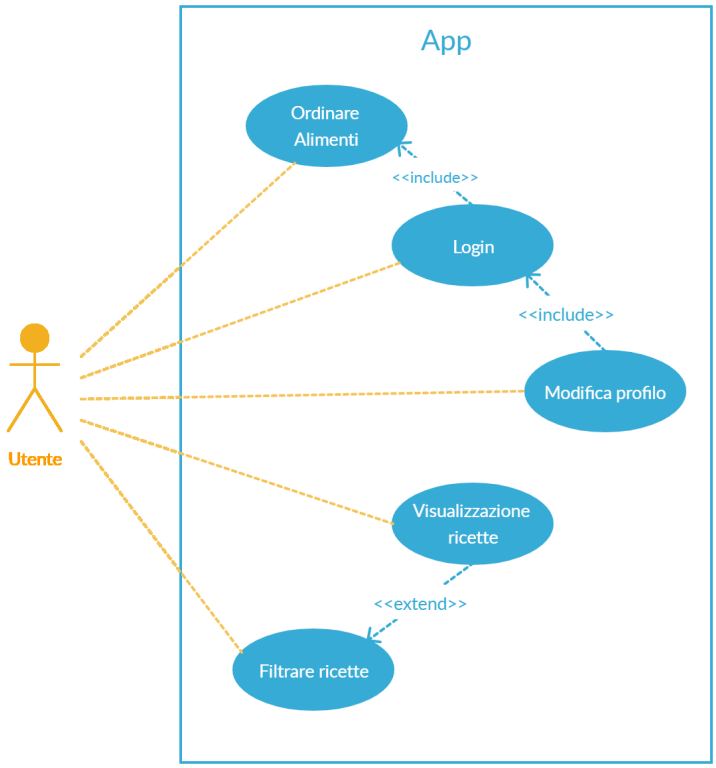


Figura 2.3: Diagramma casi d'uso dell'App EAToday

2.3.1 Spiegazione dei casi d'uso

Come si nota in Figura 2.3, i casi d'uso relativi all'applicazione sono cinque, e di seguito saranno analizzati uno per volta.

Login

Il login permette all'utente di accedere ai servizi dell'applicazione che richiedono che egli abbia effettuato il login. Come si nota in Figura 2.3, questi servizi sono due: *Ordinare alimenti* e *Modifica profilo*. Nel caso in cui il cliente utilizzi l'App per la prima volta dovrà effettuare la registrazione, in cui dovrà inserire i propri dati.

Modifica profilo

Questa funzione permette all'utente di modificare i dati del proprio profilo che, precedentemente, aveva inserito. Tale caso d'uso richiede che l'utente si sia autentificato, ed è utile nell'eventualità in cui egli voglia modificare qualche dato personale (un esempio può essere l'email).

Visualizzazione ricette

Appena l'utente apre l'applicazione, la prima cosa che visualizza è una lista di ricette selezionate in base agli alimenti a breve scadenza nei negozi vicino a lui. Tale funzionalità non richiede che l'utente abbia effettuato il login.

Filtrare ricette

Questa funzione è un caso d'uso esteso dalla precedente, in cui, però, l'utente può filtrare le ricette in base agli ingredienti. Ciò è utile nel caso in cui il cliente voglia visualizzare ricette contenenti alimenti di cui è già in possesso.

Ordinare alimenti

Tramite quest'ultima funzionalità, l'utente può ordinare gli alimenti che servono per la realizzazione di una determinata ricetta. Nell'ambito della presente tesi non viene affrontata la questione del pagamento, ma viene visualizzato solo un messaggio che conferma che non ci sono stati errori durante l'acquisto.

3

Progettazione

Questo capitolo inizia analizzando il database, soffermandosi sulla progettazione concettuale e su quella logica. In seguito, si vedrà la struttura dell'App EAToday e si mostreranno i mockup dell'applicazione.

3.1 Realizzazione del database

Le funzionalità che l'App EAToday dovrà offrire al suo utente richiedono che i dati utilizzati dall'applicazione, sia per quanto riguarda i dati personali, sia per quanto riguarda le ricette e i loro ingredienti, siano salvati in un server disponibile online. La parte di back-end dell'applicazione è stata realizzata sulla piattaforma AlterVista, che permette di costruire il database online necessario per la memorizzazione dei vari dati utilizzati dall'App.

Di seguito vengono introdotte la progettazione concettuale e quella logica utilizzate per la realizzazione del database. I dati di tale database sono, ovviamente, fittizi, in quanto servono soltanto a mostrare il comportamento dell'App. In futuro sarà compito dei vari negozi inserire nel database gli alimenti a breve scadenza che saranno, poi, utilizzati per la realizzazione delle varie ricette.

3.1.1 Progettazione concettuale

La progettazione concettuale rappresenta i dati della realtà di interesse attraverso il modello concettuale. Come modello concettuale noi utilizzeremo il modello "Entity-Relationship", E/R, che fa uso di costrutti fondamentali, quali, appunto, entità e relationship. In questa fase si individuano, dunque, le entità della realtà di interesse, e le associazioni che sussistono tra di esse. Lo scopo della progettazione concettuale è quello di effettuare una descrizione formale della realtà. Come si può notare nella Figura 3.1, lo schema E/R dell'App EAToday è molto semplice, perchè risulta composto da sole 4 entità e 2 relationship. Particolare attenzione va riposta sull'entità "Utente" che non è collegata con nessuna altra entità, in quanto, all'interno di essa, si salveranno solo i dati personali dell'utente.

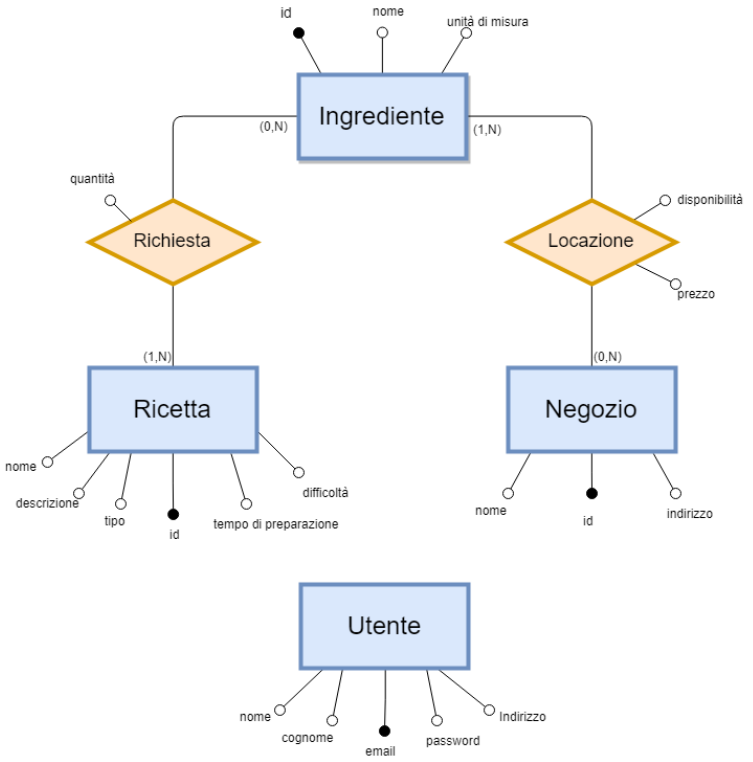


Figura 3.1: Schema Entità/Relazione

Descrizione schema E/R

I principali costrutti del modello E/R sono entità e relationship, da cui il modello prende anche il nome. Tuttavia un ruolo importante è rivestito anche dagli attributi, che esprimono proprietà elementari associate ad una entità o ad un'associazione. Di seguito si analizzano tali costrutti, uno per volta:

- *Entità*: rappresentano classi di oggetti (fatti, cose, persone, etc.) che hanno proprietà comuni ed esistenza autonoma ai fini dell'applicazione di interesse. Un'occorrenza di un'entità è un oggetto o un'istanza della classe rappresentata dall'entità. In uno schema, ogni entità viene identificata univocamente da un nome espressivo, generalmente al singolare per indicare, appunto, una classe di oggetti; essa 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 2. Per il nome dell'associazione si opta, generalmente, per un sostantivo singolare, preferibilmente un nome espressivo, che consente di evitare di dare un verso alla relationship. Di norma una relationship viene rappresentata graficamente da un rombo contenente il suo nome.

- *Attributo*: le entità e le corrispettive relationship possono essere descritte usando una serie di attributi. La scelta degli attributi riflette il livello di dettaglio con il quale si vogliono rappresentare le informazioni. Per ogni entità va scelto un identificatore, che può essere interno o esterno.

Un'altra caratteristica molto importante dello schema è la cardinalità delle relationship, che rappresenta il numero minimo e massimo di occorrenze dell'associazione cui ciascuna occorrenza di entità in essa coinvolta può partecipare.

Unitamente allo schema E/R, la progettazione concettuale prevede la presenza di un dizionario dei dati, costruito sia per le entità che per le relationship, il cui scopo è appunto quello di descrivere la realtà di interesse attraverso i costrutti scelti; ad esso si affianca spesso una dizionario dei vincoli. Il dizionario delle entità per l'App EAToday viene rappresentato nella Tabella 3.1 mentre il dizionario delle Relationship viene rappresentato nella Tabella 3.2.

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 delle entità per lo schema E/R di Figura 3.1

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 delle relationship per lo schema E/R di Figura 3.1

3.1.2 Progettazione logica

La seconda fase della progettazione di una base di dati riguarda la progettazione logica. Durante questa fase si effettua la traduzione dallo schema concettuale allo schema logico, utilizzando il modello che si preferisce. Il modello logico ha lo scopo di rappresentare le informazioni contenute nel modello concettuale in un formato più vicino alla macchina, garantendo, comunque, correttezza ed efficienza. La progettazione logica si suddivide in due fasi, ovvero la ristrutturazione dello schema E/R e la traduzione dello schema E/R ristrutturato nel modello logico prescelto, nel nostro caso quello relazionale.

Ristrutturazione dello schema E/R

La ristrutturazione si suddivide in una serie di passi da effettuare in sequenza:

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 ciascuna di esse, se risulta conveniente conservarle o, invece, rimuoverle.
2. *Eliminazione delle generalizzazioni*: tutte le generalizzazioni dello schema vanno sostituite da altri costrutti perchè il modello relazionale non permette la loro rappresentazione. Ci sono tre modi per eliminare le generalizzazioni, ovvero:
   - *Accorpamento delle entità figlie nell'entità padre*: conveniente quando l'accesso è contestuale, ossia quando le operazioni non fanno molta distinzione tra occorrenze di padre e figli.
   - *Accorpamento dell'entità padre nelle entità figlie*: possibile solo se la generalizzazione è totale (altrimenti alcune occorrenze del padre non verrebbero rappresentate). Conveniente quando le operazioni si riferiscono solo a una specifica entità figlia.
   - *Sostituzione delle generalizzazioni con associazioni*: possibile, e conveniente, anche quando la generalizzazione è parziale e ci sono operazioni solo su una delle figlie o sul padre.
3. *Partizionamento/accorpamento di entità e 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; più specificatamente:
   - gli attributi con valori nulli non possono costituire identificatori principali;
   - un identificatore composto da uno o pochi attributi è da preferire ad un identificatore composto da molti attributi;
   - un identificatore interno è da preferire ad uno esterno;
   - si preferisce un identificatore utilizzato da molte operazioni per accedere alle occorrenze di una entità.

Traduzione dello schema E/R

Poichè lo schema concettuale è molto semplice non è stato necessario effettuare alcuna ristrutturazione; per questo motivo, è stato utilizzato lo stesso schema della Figura 3.1 per ottenere lo schema relazionale. Tutte le relazioni, presenti nello schema E/R sono di tipo "molti a molti"; per tale ragione, per quanto riguarda le

associazioni, si utilizza una relazione avente come attributi i suoi attributi e come identificatore le chiavi delle entità coinvolte (che costituiscono la chiave primaria). Applicando tali regole si ottiene il seguente schema relazionale:

- **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)

3.2 Struttura dell'App

La struttura in Figura 3.2 specifica come è organizzata l'applicazione e quali sono i passaggi da fare per raggiungere una determinata pagina. Il suo scopo è quello di aiutare l'utente nell'utilizzo dell'App.

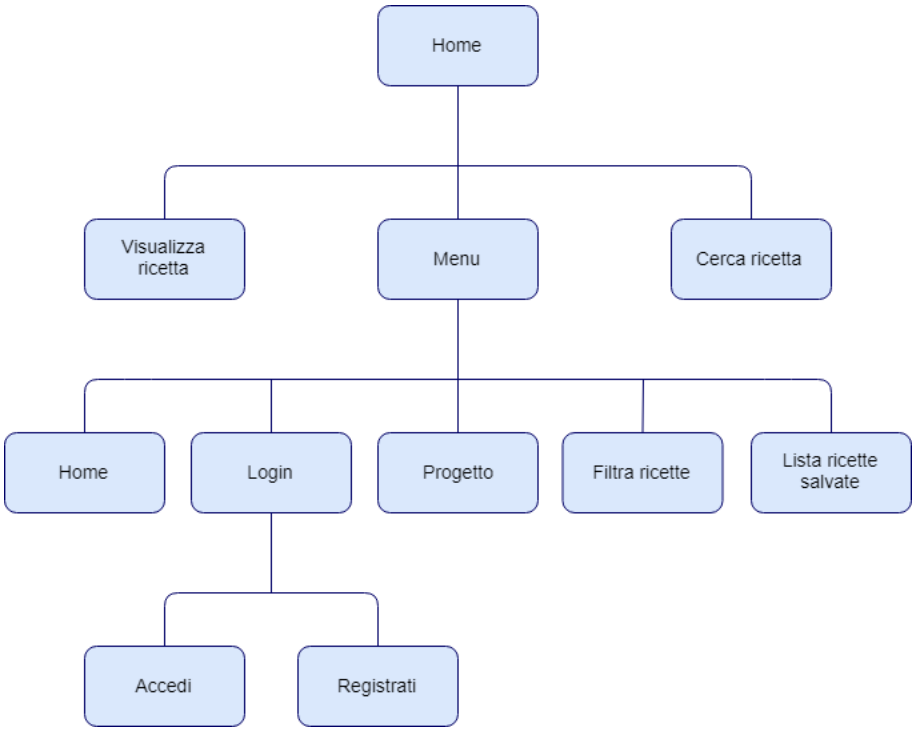


Figura 3.2: Struttura dell'App EAToday

All'avvio dell'applicazione le varie ricette vengono scaricate dal database, grazie a uno script PHP, e successivamente mostrate all'utente sul display. Come si può

notare dalla Figura 3.2, l'utente, dalla schermata "Home", può eseguire le seguenti operazioni:

- *Cerca Ricetta*: l'utente, attraverso la barra di ricerca, 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, premendo sulla casella di ogni ricetta, può visualizzare i suoi ingredienti e come prepararla. In tale pagina egli ha anche la possibilità di acquistare gli ingredienti di cui ha bisogno e salvare la ricetta nel caso la volesse rivedere.
- *Menù*: l'utente apre il menù grazie all'apposito pulsante (Menù); così facendo, come si nota nella Figura 3.3, egli ha la possibilità di effettuare altre operazioni, tra cui effettuare l'accesso (in caso di primo accesso, l'utente si deve registrare), filtrare le ricette, visualizzare le ricette precedentemente salvate, visualizzare le caratteristiche del progetto, ovvero la spiegazione delle funzionalità dell'app e, infine, tornare alla "Home".

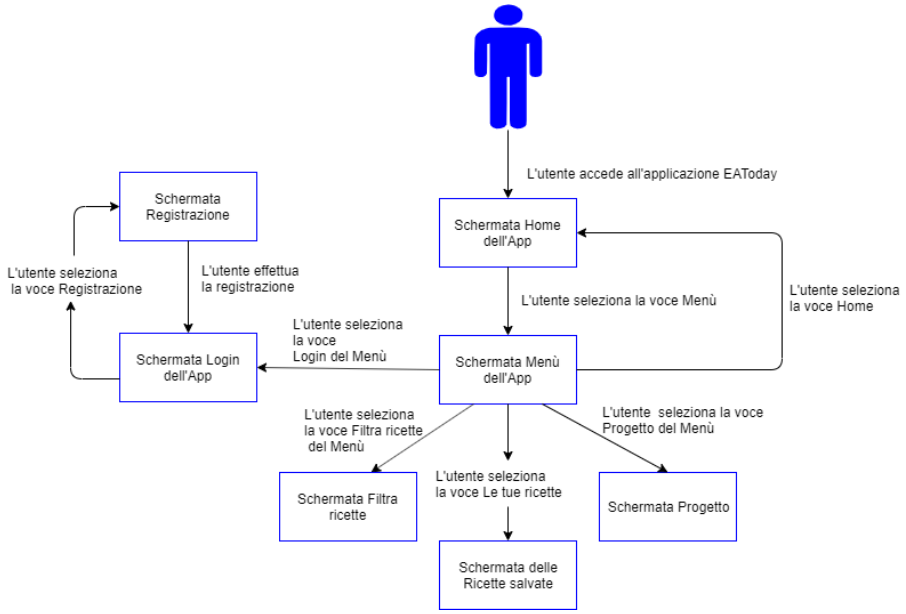


Figura 3.3: Funzionamento della schermata relativa al Menù nel caso in cui l'utente non abbia effettuato il Login

Se l'utente ha effettuato il Login, il Menù cambia e le operazioni possibili diventano quelle in Figura 3.4; come si evince dalla figura, la voce "Login" viene sostituita dalla voce "Profilo" dell'utente e si ha anche la possibilità di effettuare il Logout. Quando l'utente effettua il Logout la schermata del "Menù" ritorna ad essere quella rappresentata nella Figura 3.3.

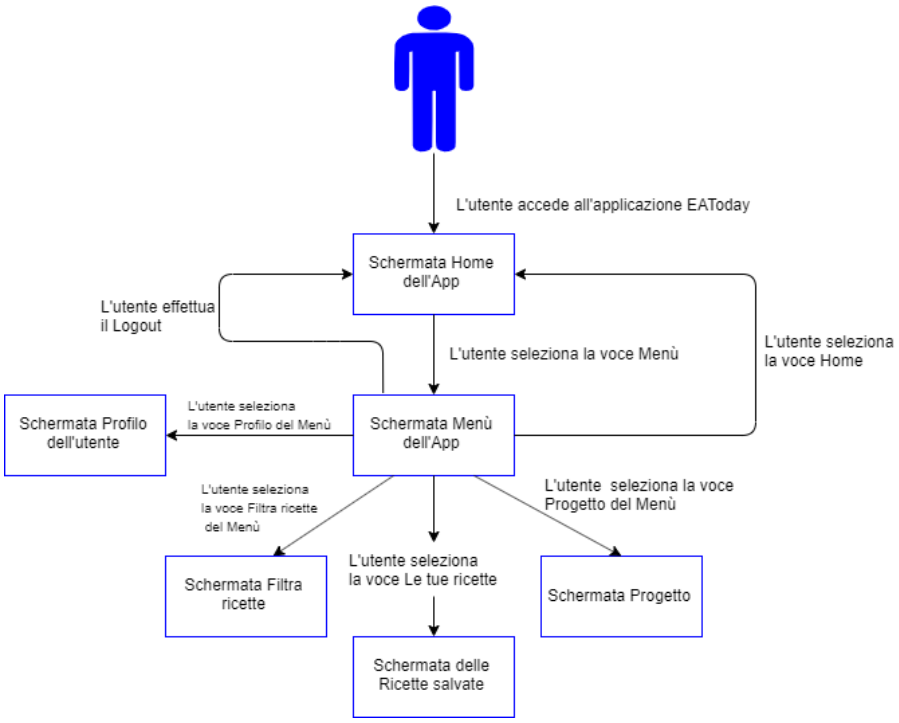


Figura 3.4: Funzionamento della schermata relativa al Menù nel caso in cui l'utente abbia effettuato il Login

3.3 Mockup

I mockup sono uno strumento della progettazione della componente applicativa molto importante; essi, infatti, danno all'utente un'idea di come sarà il risultato finale del prodotto dal punto di vista grafico. In realtà, prima di rappresentare il mockup finale, esistono i cosìdetti "mockup di livello 0" (wireframe) che sono molto più semplici.

Il wireframe è una bozza grafica, uno schizzo che rappresenta la base dell'idea. Solitamente è in bianco e nero perchè in questa fase l'attenzione è focalizzata sugli elementi che caratterizzano il proprio prodotto, come, nel caso di un'App, la posizione dei vari elementi nelle varie schermate.

Per quanto riguarda i mockup finali (detti di "livello 2"), essi rappresentano come realmente saranno disposti i vari elementi nelle diverse schermate tenendo in forte considerazione gli aspetti relativi alla grafica, a differenza dei wireframe, che, invece, danno un'idea molto più approssimata sia della disposizione che grafica.

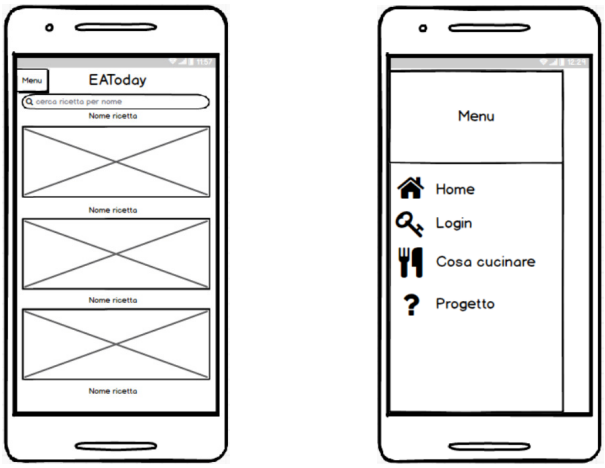
Di seguito vengono rappresentati i wireframe dell'App EAToday; i mockup finali non saranno presentati perchè coincidono con le schermate dell'App realizzate che verranno illustrate nel prossimo capitolo.

Schermata relativa alla home page

Il wireframe in Figura 3.5(a) rappresenta la schermata iniziale dell'applicazione EAToday. In modo semplice vengono rappresentate le ricette, il pulsante menù, la barra di ricerca e il nome dell'applicazione.

Schermata del menù

Il wireframe in Figura 3.5(b) rappresenta come sarà la schermata una volta che l'utente ha premuto il tasto "Menù". Si può osservare che l'utente avrà a disposizione quattro possibili scelte nel menù.



(a) *"Home"* (b) *"Menù"*

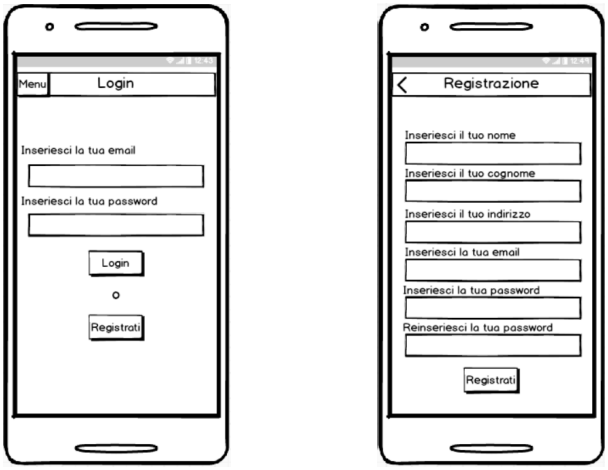
Figura 3.5: Wireframe delle schermate relative alla Home Page e al Menu

Schermata Login

Il wireframe in Figura 3.6(a) rappresenta la schermata del Login. L'utente ha la possibilità di inserire i propri dati oppure può registrarsi, nel caso in cui non l'abbia già fatto.

Schermata della Registrazione

Il wireframe in Figura 3.6(b) rappresenta la schermata della registrazione. L'utente dovrà inserire i vari dati affinchè venga effettuata la registrazione.



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

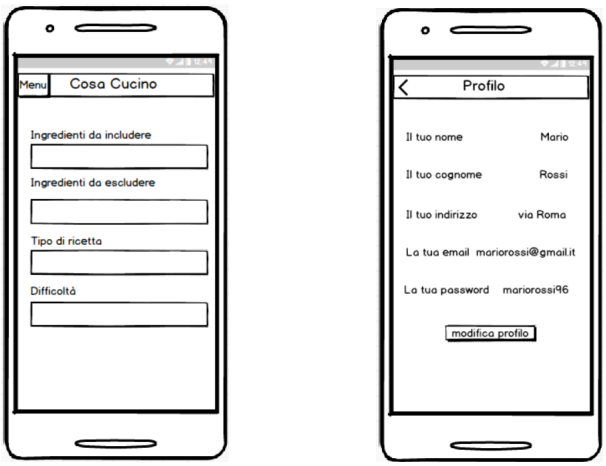
Figura 3.6: Wireframe delle schermate relative al Login e alla registrazione

Schermata Cosa cucino

Il wireframe in Figura 3.7(a) rappresenta la schermata quando l'utente sceglie la voce "Cosa cucino" nel menù. Si può osservare che all'utente sarà richiesto che ingredienti intende includere o escludere dalla ricetta, che tipologia di ricetta intende fare e che difficoltà essa dovrà avere.

Schermata Profilo

Il wireframe in Figura 3.7(b) rappresenta la schermata del Profilo di un utente.



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

Figura 3.7: Wireframe delle schermate profilo e filtro

4

Implementazione e manuale utente

Questo capitolo tratta l'implementazione dell'App EAToday. Inizialmente si analizzano gli strumenti utilizzati per lo sviluppo; successivamente, viene mostrato e spiegato il codice utilizzato per la realizzazione delle varie schermate dell'applicazione.

4.1 Strumenti utilizzati

4.1.1 Microsoft Visual Studio

Per la realizzazione dell'applicazione Xamarin EAToday è stato utilizzato l'IDE Visual Studio. Microsoft Visual Studio è un ambiente di sviluppo integrato (Integrated Development Environment o IDE) fornito da Microsoft, che supporta diversi linguaggi di programmazione, tra cui C, C++, C#, Visual Basic .Net e JavaScript. Grazie all'utilizzo di questi ultimi, Visual Studio permette la creazione di diversi tipi di software tra cui applicazioni desktop, applicazioni web e mobile App.

Visual Studio utilizza la tecnologia "IntelliSense", che permette di correggere eventuali errori sintattici senza compilare l'applicazione, e possiede anche un debugger interno per aiutare il rilevamento e la correzione degli errori logici nel codice in *runtime*.

4.1.2 C#

C# è un linguaggio di programmazione orientato agli oggetti sviluppato da Microsoft. C# è un linguaggio sia interpretato che compilato, e la trasformazione in linguaggio macchina avviene durante l'esecuzione del programma; inizialmente il codice sorgente viene convertito in un codice intermedio detto "Intermediate Language"(IL), e solo a *runtime* esso viene trasformato in linguaggio macchina adatto per l'hardware su cui deve funzionare.

4.1.3 AlterVista

AlterVista è una piattaforma web dove è possibile aprire gratuitamente un sito web. È stata fondata nel 2000 da uno studente del Politecnico di Torino.

Su *AlterVista* 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 delle pubblicità.

La principale utilità di AlterVista per l'App EAToday è stata sostanzialmente quella di fornire degli *Script PHP* per tradurre le informazioni contenute nel database in una struttura dati *JSON*. Tramite l'utilizzo di richieste *GET* e *POST* è stato possibile ricevere dati e inviarli al database in modo da far si che il funzionamento dell'applicazione fosse corretto.

PhpMyAdmin consente di realizzare un database da zero, permettendo di eseguire varie operazioni sulle tabelle di quest'ultimo. Sono, anche, previste delle funzionalità per l'inserimento dei dati (utili per il popolamento del database), per le query e per il backup dei dati.

4.1.4 PHP

PHP (acronimo ricorsivo di " PHP: Hypertext Preprocessor") è un linguaggio di scripting interpretato, originariamente concepito per la programmazione di pagine web dinamiche. L'interprete PHP è un software libero distribuito sotto la PHP License.

Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server, ma può essere usato anche per realizzare script a riga di comando o applicazioni stand-alone con interfaccia grafica.

PHP riprende per molti versi la sintassi del C, come fanno molti linguaggi moderni, e del Perl. Esso è un linguaggio a tipizzazione debole e dalle versioni più recenti migliora il supporto al paradigma di programmazione ad oggetti.

PHP è in grado di interfacciarsi a innumerevoli DBMS, tra cui MySQL, PostgreSQL, MariaDB, Oracle, Firebird, IBM DB2, Microsoft SQL Server e anche database NoSQL come MongoDB.

4.1.5 JSON

JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, che utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, *Java*, *JavaScript*, *Python*, e molti altri. Questa caratteristica fa si che JSON sia 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, uno 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, supportate da tutti i linguaggi di programmazione moderni.

4.1.6 XAML

XAML è un linguaggio di markup basato su XML, utilizzato per descrivere l'interfaccia grafica delle applicazioni basate sulla libreria "Windows Presentation Foundation". XAML si basa sugli oggetti contenuti nel Common Language Runtime e sulle loro proprietà o eventi.

4.2 Implementazione back-end dell'App EAToday

Affinchè l'applicazione funzioni correttamente, come è stato già detto, essa si deve connettere al database per recuperare i vari dati. Per fare questo, nella componente back-end sono stati realizzati degli script in PHP che permettono all'utente di effettuare il login, registrarsi e scaricare le varie ricette. Nel seguito vengono mostrati tali script.

Login

Affinchè l'utente possa effettuare il login sull'applicazione è stato realizzato lo script mostrato nel Listato 4.1 che riceve, dalla richiesta del front-end, l'email e la password inserita dall'utente e verifica se tali dati sono presenti all'interno del database. Se sono presenti, l'utente, nel lato front-end, effettua il login e i suoi dati vengono scaricati, altrimenti viene mostrato un messaggio di errore.

```
1  <?php
2  $json = file_get_contents ('php://input');
3  $data = json_decode($json);
4
5  $email = $data->email;
6  $password = $data->password;
7
8
9  require "../connection.php";
10
11 if ($email != null && $password != null){
12
13 $query = "select * from user where email = '".$email."' and password
       ='".$password."';";
14 $result = mysqli_query($connection,$query);
15
16 if (mysqli_num_rows($result)> 0){
17   $response = array();
18     $code = "login_true";
19     $row = mysqli_fetch_array($result);
20     $message = "Login success";
```

```
21      array_push($response,array("code"=>$code,"message"=>$message));
22      echo json_encode(array("user" => array("name" =>
23      $row[0],"lastName" =>$row[1],"email" =>$row[2],"password" =>
24      $row[3],"address" =>$row[4]),"server_response"=>$response));
25  }
26  else{
27    $response = array();
28      $code = "login_false";
29      $row = mysqli_fetch_array($result);
30      $name = $row[0];
31      $message = "Login failed... Try again";
32      array_push($response,array("code"=>$code,"message"=>$message));
33      echo json_encode(array("server_response"=>$response));
34  }
35  }
36      mysqli_close($connection);
37
38  ?>
```

Listato 4.1: Script PHP utilizzato per effettuare il Login

Registrazione

Un nuovo utente ha la possibilità di registrarsi grazie allo script mostrato nel Listato 4.2. Esso riceve i dati inseriti dall'utente nell'applicazione e li salva all'interno del database. Così facendo l'utente può successivamente effettuare il login.

```
1  <?php
2  $json = file_get_contents ('php:');
3  $data = json_decode($json);
4
5  $name = $data->name;
6  $lastName = $data->lastName;
7  $email = $data->email;
8  $password = $data->password;
9  $address = $data->address;
10
11 require "../connection.php";
12
13 $query = "select * from user where email= '$email';";
14 $result  = mysqli_query($connection,$query);
15
16 if($email != null  && $password != null && $name != null){
17
18 if  (mysqli_num_rows($result) > 0){
19   $response = array();
20      $code = "register_false";
21      $message = "User already Exist...";
22      array_push($response,array("code"=>$code,"message"=>$message));
23      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 4.2: Script PHP utilizzato per effettuare la registrazione

Scarica ricette

Lo script mostrato nel Listato 4.3 permette all'applicazione di scaricare le ricette che saranno mostrate all'utente. È stato realizzato in modo da riuscire a implementare anche il filtraggio di ricette; se l'utente non ha inserito alcun filtro vengono mostrate tutte le ricette disponibili, altrimenti vengono mostrate solo quelle che rispettano i requisiti.

```
<?php
$recipe_name = urldecode($_GET['name']);
$type = $_GET['type'];
 $difficulty  = $_GET['difficulty '];
$ingredients_s  = $_GET['selected'];
$ingredients_e  = $_GET['excluded'];

require "../connection.php";
header('Content-Type:Application/json');
```

```
10
11 function query_from_string($query, $con){
12   $result_recipes = mysqli_query($con,$query);
13     while ($row = mysqli_fetch_assoc( $result_recipes )) {
14     $array_recipes [] = $row;
15     }
16     foreach ( $array_recipes as &$recipe){
17         $query_join = "SELECT * FROM collection INNER JOIN ingredient
                ON collection.id_ingredient =
18         ingredient.id WHERE collection.id_recipe = '".$recipe['id']."';";
19          $result_collection = mysqli_query($con,$query_join);
20         while ($row = mysqli_fetch_assoc( $result_collection )) {
21         $array_ingredients [] = $row;
22       }
23         $recipe [' ingredients '] = $array_ingredients;
24         unset($array_ingredients);
25     }
26     return $array_recipes;
27 }
28
29 if ($recipe_name == null && $type == null && $difficulty == null){
30     $query_recipes = "SELECT * FROM recipe";
31     $array_recipes = query_from_string($query_recipes, $connection);
32 }else{
33   $string_query = "SELECT * FROM recipe WHERE ";
34   if ($recipe_name != null){
35     $string_query = "".$string_query."name = '".$recipe_name."'";
36   }else{
37     if ($type != null && $difficulty != null){
38           if($type == "vegetariano"){
39                $string_query = "".$string_query."(type = 'vegetariano' OR
                      type = 'vegano')
40               AND difficulty = '".$difficulty."'";
41             }else{
42                $string_query = "".$string_query."type = 'vegano' AND
                     difficulty = '".$difficulty."'";
43             }
44       }else if ( $difficulty != null){
45             $string_query = "".$string_query."difficulty = '".$difficulty."'
                  ";
46         }else if ($type != null){
47           if($type == "vegetariano"){
48                $string_query = "".$string_query."(type = 'vegetariano' OR
                      type = 'vegano')";
49             }else{
50                $string_query = "".$string_query."type = 'vegano'";
51             }
52         }
53         $string_query = "".$string_query.";";
54         $array_recipes = query_from_string($string_query, $connection);
55     }
```

```
56 }
57
58 $check = 0;
59 if ($ingredients_s != null){
60   $array = $array_recipes;
61   unset($array_recipes);
62    $ingredients_selected = explode(",",$ingredients_s);
63     foreach ( $array as $recipe){
64       $ingredients = $recipe['ingredients'];
65         foreach( $ingredients as $in){
66       $array_tmp[] = $in['id'];
67           }
68           foreach( $ingredients_selected as $in){
69        if (!in_array($in,$array_tmp)){
70            $check = 1;
71                   break;
72            }
73           }
74           if ($check == 0){
75            $array_recipes [] = $recipe;
76           }
77          unset($array_tmp);
78          $check = 0;
79          }
80     unset($array);
81 }
82
83 if ($ingredients_e != null){
84   $array = $array_recipes;
85   unset($array_recipes);
86   $ingredients_excluded = explode(",",$ingredients_e);
87     foreach ( $array as $recipe){
88       $ingredients = $recipe['ingredients'];
89         foreach( $ingredients as $in){
90       $array_tmp[] = $in['id'];
91           }
92           foreach( $ingredients_excluded as $in){
93        if (in_array($in,$array_tmp)){
94            $check = 1;
95                   break;
96            }
97           }
98           if ($check == 0){
99            $array_recipes [] = $recipe;
100          }
101         unset($array_tmp);
102         $check = 0;
103         }
104 }
105
106
```

```
107 echo json_encode($array_recipes);
108
109
110 mysqli_close($connection);
111 ?>
```

Listato 4.3: Script PHP utilizzato per scaricare le ricette

4.3 Implementazione front-end dell'App EAToday

Per quanto riguarda l'organizzazione del codice abbiamo usato il pattern software MVVM (Model-View-ViewModel), che separa gli aspetti dell'applicazione in tre componenti:

- *Model*: rappresenta il punto di accesso ai dati.
- *View*: rappresenta l'interfaccia grafica che mostrerà i dati.
- *ViewModel*: rappresenta il punto di incontro tra la View e il Model.

Il fulcro del funzionamento del pattern è la creazione del ViewModel, che rappresenta tutte le informazioni e i componenti della corrispondente View. Quest'ultima, infatti, si limita a rappresentare graficamente quanto esposto dal ViewModel.

In Figura 4.1 viene rappresentato come è stato strutturato il front-end dell'applicazione.

Prima di iniziare a spiegare le schermate più importanti è fondamentale spiegare la classe `ApiServices` che è situata all'interno della cartella `Services`, dentro il Modem. È importantissima perchè essa permette all'App di fare i collegamenti con il database online. Come si vede nel Listato 4.4 la classe `ApiServices` ha 4 metodi:

- `RegisterAsync()`: metodo che viene chiamato quando l'utente ha intenzione di registrarsi per la prima volta. Esso riceve i dati inseriti dall'utente (nome, cognome, email, etc.), li serializza in un JSON ed effettua una richiesta POST ad AlterVista. Se la registrazione avviene con successo l'utente può effettuare il login.
- `LoginAsyn()`: metodo che viene chiamato quando l'utente ha intenzione di effettuare il login. Si prendono l'email e la password inseriti dall'utente, vengono serializzati, e viene effettuata una richiesta POST; se tali valori sono corretti e presenti nel database l'utente effettua il login correttamente, altrimenti viene mostrato un messaggio di errore. Quando si effettua il login vengono anche scaricati i dati dell'utente e salvati in locale; se l'utente lo volesse, avrebbe anche la possibilità di modificarli.
- `RecipeAsync()`: metodo chiamato all'avvio dell'applicazione che, grazie ad una richiesta GET, fa si che le ricette vengano scaricate; successivamente, grazie al ViewModel e alla View, le ricette vengono mostrate all'utente.
- `IngredientAsync()`: metodo utilizzato per scaricare tutti gli ingredienti, grazie a una richiesta GET, per facilitare l'utente a capire in base a quali di questi può filtrare le ricette.

Detto ciò possiamo analizzare più in dettaglio ogni schermata.

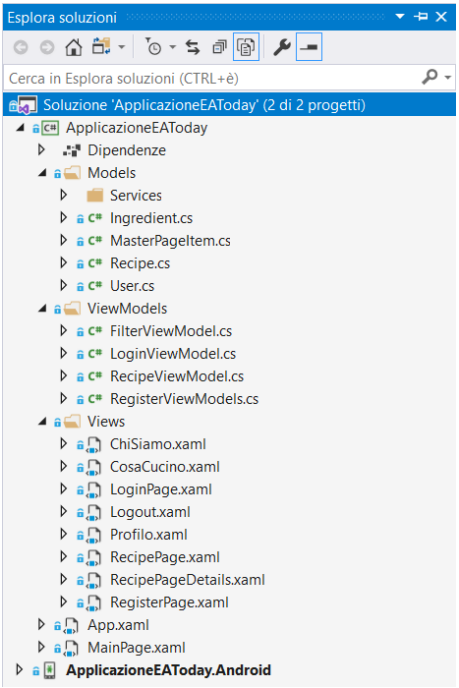


Figura 4.1: Componenti del pattern dell'applicazione EAToday

```
1  using System.Threading.Tasks;
2  using System.Net.Http;
3  using ApplicazioneEAToday.Models;
4  using Newtonsoft.Json;
5  using System.Net.Http.Headers;
6  using System.Collections.Generic;
7  using System.Diagnostics;
8  using System.Collections.ObjectModel;
9  using System;
10 using Plugin.Connectivity;
11 using Newtonsoft.Json.Linq;
12 using ApplicazioneEAToday.Views;
13
14 namespace ApplicazioneEAToday.Services
15 {
16     public class ApiServices
17     {
18
19         public async Task<bool> RegisterAsync(string email,
20         string name,string lastName, string address, string password)
21         {
22             var client = new HttpClient();
23             User user=new User(name,lastName,email,password,address);
24             var json = JsonConvert.SerializeObject(user);
```

```
25              Debug.WriteLine(json);
26              HttpContent content = new StringContent(json);
27              content.Headers.ContentType = new MediaTypeHeaderValue("
                    application/json");
28              var response = await client.PostAsync("http://eatoday.
                    altervista.org/access/register.php", content);
29              var responseContent = await response.Content.
                    ReadAsByteArrayAsync();
30              Debug.WriteLine(System.Text.Encoding.UTF8.GetString(
                    responseContent));
31              return response.IsSuccessStatusCode;
32          }
33          public async Task<bool> LoginAsync(string email, string password)
34          {
35              var client = new HttpClient();
36              var values = new Dictionary<string, string>();
37              values.Add("email", email);
38              values.Add("password", password);
39              var loginJson = JsonConvert.SerializeObject(values);
40              Debug.WriteLine(loginJson);
41              HttpContent content = new StringContent(loginJson);
42              content.Headers.ContentType = new MediaTypeHeaderValue("
                    application/json");
43              var response = await client.PostAsync("http://eatoday.
                    altervista.org/access/login.php", content);
44              var responseContent = await response.Content.
                    ReadAsByteArrayAsync();
45              string json = System.Text.Encoding.UTF8.GetString(
                    responseContent);
46              Debug.WriteLine(System.Text.Encoding.UTF8.GetString(
                    responseContent));
47
48              if (response.IsSuccessStatusCode)
49              {
50                  JObject jsonObject = JObject.Parse(json);
51                  JArray jsonResponse = (JArray)jsonObject.SelectToken("
                        server_response");
52                  JObject jsonResp = (JObject)jsonResponse[0];
53                  string code = (string)jsonResp.SelectToken("code");
54                  string message = (string)jsonResp.SelectToken("message");
55                  if (code.Equals("login_true"))
56                  {
57                      JObject jsonUser = (JObject)jsonObject.SelectToken("
                            user");
58                      var s = jsonUser.ToString();
59                      User user = new User();
60                      user = JsonConvert.DeserializeObject<User>(s);
61                      Profilo.user = user;
62                      return true;
63                  }
64                  else return false;
```

```
65              }
66              else
67              {
68                  await App.Current.MainPage.DisplayAlert("Errore di
                        connessione", "", "Connetti ad una rette");
69                  return false;
70              }
71          }
72
73          public async Task<ObservableCollection<Recipe>> RecipeAsync()
74          {
75              ObservableCollection<Recipe> listaRecipe = null;
76              if (CrossConnectivity.Current.IsConnected)
77              {
78                  try
79                  {
80                      var client = new HttpClient();
81                      var content = await client.GetStringAsync("http://
                            eatoday.altervista.org/filter
82                      /recipe.php"+ CosaCucino.get);
83                      CosaCucino.get = "";
84                      var tr = JsonConvert.DeserializeObject<List<Recipe
                            >>(content);
85                      ObservableCollection<Recipe> trends = new
                            ObservableCollection<Recipe>(tr);
86                      listaRecipe = new ObservableCollection<Recipe>(trends
                            );
87                  }
88                  catch (Exception ex)
89                  {
90                      Debug.WriteLine(ex.Message.ToString());
91                  }
92                  return listaRecipe;
93              }return listaRecipe;
94
95          }
96
97          public async Task<ObservableCollection<Ingredient>>
                IngredientAsync()
98          {
99              ObservableCollection<Ingredient> listaIngredient = null;
100
101             if (CrossConnectivity.Current.IsConnected)
102             {
103                 try
104                 {
105                     var client = new HttpClient();
106                     var content = await client.GetStringAsync("http://
                            eatoday.altervista.org/filter/ingredient.php")
                            ;
107                     var tr = JsonConvert.DeserializeObject<List<Ingredient
```

```
                    >>(content);
108                 ObservableCollection<Ingredient> trends = new
                        ObservableCollection<Ingredient>(tr);
109                 listaIngredient  = new ObservableCollection<Ingredient
                        >(trends);
110             }
111             catch (Exception ex)
112             {
113                 Debug.WriteLine(ex.Message.ToString());
114             }
115             return listaIngredient ;
116         }
117         return listaIngredient ;
118     }
119 }
120 }
```

Listato 4.4: Classe `ApiServices`

4.3.1 MasterDetailPage

MasterDetailPage è una pagina che gestisce due pagine correlate di informazioni, ovvero una pagina master, che presenta gli elementi, e una pagina di dettaglio, che presenta i dettagli relativi agli elementi della pagina master.

Implementazione della MasterDetailPage

Come si può notare nel Listato 4.5 viene creato un oggetto "menuList" di tipo `List<MasterPageDettail>`, e, in base al fatto che l'utente abbia effettuato o meno il login, vengono aggiunti elementi visibili all'utente sul display. `Detail = new NavigationPage((Page)Activator.CreateInstance(typeof(RecipePage)))` fa si che all'avvio la pagina Master è impostata sulla RecipePage.

Il metodo `OnMenuItemSelected()` spiega cosa accadde quando viene selezionato un elemento della MasterDetailPage.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using Xamarin.Forms;
7  using ApplicazioneEAToday.Views;
8  using ApplicazioneEAToday.Models;
9
10 namespace ApplicazioneEAToday
11 {
12     public partial class MainPage : MasterDetailPage
13     {
14         public List<MasterPageItem> menuList { get; set; }
```

```
15          public MainPage()
16          {
17              InitializeComponent();
18              Title = "EAToday";
19              menuList = new List<MasterPageItem>();
20
21              menuList.Add(new MasterPageItem() { Title = "Home",
22                  TargetType = typeof(RecipePage) });
23
24              if (User.log == false)
25              {
26                  menuList.Add(new MasterPageItem() { Title = "Login",
27                      TargetType = typeof(LoginPage) });
28              }else
29              {
30                  menuList.Add(new MasterPageItem() { Title = "Profilo",
31                      TargetType = typeof(Profilo) });
32
33              }
34              menuList.Add(new MasterPageItem() { Title = "Cosa Cucino",
35                  Icon = "icon.png",
36                  TargetType = typeof(CosaCucino) });
37              menuList.Add(new MasterPageItem() { Title = "Chi Siamo",
38                  TargetType = typeof(ChiSiamo) });
39
40              if (User.log == true)
41              {
42                  menuList.Add(new MasterPageItem() { Title = "Logout",
43                      TargetType = typeof(Logout) });
44              }
45
46              navigationDrawerList.ItemsSource = menuList;
47              Detail = new NavigationPage((Page)Activator.CreateInstance(
                    typeof(RecipePage)));
48          }
49          private void OnMenuItemSelected(object sender,
                SelectedItemChangedEventArgs e)
50          {
51              var item = (MasterPageItem)e.SelectedItem;
52              Type page = item.TargetType;
53              Detail = new NavigationPage((Page)Activator.CreateInstance(
                    page));
54              IsPresented = false;
55          }
56      }
57  }
```

Listato 4.5: Implementazione della MasterDetailPage

Implementazione grafica MasterDetailPage

Nel Listato 4.6 viene impostato il modo in cui devono essere rappresentati gli elementi creati nel Listato 4.5. `ListView x:Name="navigationDrawerList"` fa si che gli elementi della MasterDetailPage siano quelli creati in precedenza; `ItemSelected= "OnMenuItemSelected"` indica che, quando viene selezionato un elemento, deve partire il metodo `OnMenuItemSelected()`.

```
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
                  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                  x:Class="ApplicazioneEAToday.MainPage">

    <MasterDetailPage.Master>
        <ContentPage BackgroundColor="#2196F3" Title="Menu">
            <Grid BackgroundColor="Transparent">
                <Grid.RowDefinitions>
                    <RowDefinition Height="200" />
                    <RowDefinition Height="*" />
                </Grid.RowDefinitions>
                <Grid>

                    <StackLayout Padding="0,20,0,0"
                                 HorizontalOptions="CenterAndExpand"
                                 VerticalOptions="CenterAndExpand">

                        <Label Text="Menu"
                               TextColor="White"
                               FontSize="Large" />

                    </StackLayout>

                </Grid>

                <StackLayout Grid.Row="1" Spacing="15">
                    <ListView x:Name="navigationDrawerList"
                        RowHeight="60"
                        SeparatorVisibility ="None"
                        BackgroundColor="White"
                        ItemSelected="OnMenuItemSelected">

                        <ListView.ItemTemplate >
                            <DataTemplate >
                                <ViewCell>
                                    <StackLayout >

                                        <StackLayout VerticalOptions="
                                            FillAndExpand"
                                            Orientation="Horizontal"
                                            Padding="20,10,0,10"
                                            Spacing="20">
```

```
43                                          <Label Text="{Binding
                                                Title}"
44                                          FontSize="Medium"
45                                          VerticalOptions="Center"
46                                          TextColor="#2196F3"/>
47                                      </StackLayout>
48                                      <BoxView HeightRequest="1"
49                                               BackgroundColor="Gray"
                                                    />
50
51                                  </StackLayout>
52
53                              </ViewCell>
54                          </DataTemplate>
55                      </ListView.ItemTemplate>
56                  </ListView>
57              </StackLayout>
58          </Grid>
59      </ContentPage>
60  </MasterDetailPage.Master>
61
62  <MasterDetailPage.Detail>
63      <NavigationPage>
64
65      </NavigationPage>
66  </MasterDetailPage.Detail>
67
68 </MasterDetailPage>
```

Listato 4.6: Implementazione del menù

4.3.2 RecipePage

All'avvio dell'applicazione la schermata principale è quella della Recipe Page nella quale all'utente vengono mostrate le ricette.

Implementazione della Recipe Page

Le ricette vengono caricate grazie al metodo `OnGetList()` che a sua volta richiama il metodo `CreazioneLista()`, situato in RecipeViewModel. Le ricette vengono inserite in `ricette = ObservableCollection<Recipe>`.

L'utente ha la possibilità di cercare una ricetta per nome, attraverso la barra di ricerca, grazie al metodo `SearchBar_TextChanged()` (Listato 4.7). In base alle lettere che l'utente inserisce, le ricette si aggiornano; ovviamente se non è presente nessuna ricetta il cui nome inizia con quelle lettere non viene mostrato niente.

Oltre a cercare per nome, grazie al metodo `MyListItemClick()`, l'utente può visualizzare i dettagli di una ricetta voluta.

```
1  namespace ApplicazioneEAToday.Views
2  { [XamlCompilation(XamlCompilationOptions.Compile)]
3    public partial class RecipePage : ContentPage
4    {
5          RecipeViewModel recipeVM= new RecipeViewModel();
6          FilterViewModel filterVM = new FilterViewModel();
7          public ObservableCollection<Recipe> ricette;
8          public static ObservableCollection<Ingredient>
               listaCompletaIngredienti { get; set; }
9          public RecipePage ()
10     {
11       InitializeComponent ();
12             Title = "EAToday";
13             OnGetList();
14         }
15         public async void OnGetList()
16         {
17             ricette = await recipeVM.CreazioneLista();
18             listaCompletaIngredienti = await filterVM.CreazioneLista();
19             myList.ItemsSource = ricette;
20         }
21         private  void SearchBar_TextChanged(object sender,
               TextChangedEventArgs e)
22         {
23             if (string.IsNullOrEmpty(e.NewTextValue)) myList.ItemsSource
                   = ricette;
24             else
25             {
26                 myList.ItemsSource= ricette.Where(x => x.name.StartsWith
                       (e.NewTextValue));
27             }
28         }
29         private async void MyListItemClick(object sender,
               ItemTappedEventArgs e)
30         {
31             var recipe = e.Item as Recipe;
32             var vm = BindingContext as RecipeViewModel;
33             List<Ingredient> list = recipe.ingredients;
34             await Navigation.PushAsync(new RecipePageDetails(recipe.name,
                   recipe.description,
35                 recipe.imageURL, list));
36         }
37     }
38 }
```

Listato 4.7: Implementazione della schermata delle ricette

Implementazione grafica della Recipe Page

Nel Listato 4.8 viene mostrato come le varie ricette devono essere rappresentate. Le ricette vengono caricate grazie al comando `ListView x:Name="myList"` (`myList`

viene caricato grazie al comando `myList.ItemsSource = ricette` che si trova nel Listato 4.7).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="ApplicazioneEAToday.Views.RecipePage"
             xmlns:vm="clr-namespace:ApplicazioneEAToday.ViewModels"
                 >

    <ContentPage.BindingContext>
        <vm:RecipeViewModel/>
    </ContentPage.BindingContext>
    <StackLayout>
        <SearchBar TextChanged="SearchBar_TextChanged"
              Placeholder="Cerca per nome"/>
        <ListView x:Name="myList"
                  RowHeight="100"
                  HasUnevenRows="true"
                  ItemTapped="MyListItemClick" >

            <ListView.ItemTemplate>

                <DataTemplate>

                    <ViewCell >


                                <StackLayout
                                    Margin="5,5,5,5"
                                    HorizontalOptions="
                                        CenterAndExpand"
                                    Orientation="Vertical" >

                                    <Label
                                        FontSize="Subtitle"
                                        Text="{Binding name}"
                                        VerticalOptions="
                                            CenterAndExpand"
                                        HorizontalOptions="
                                            CenterAndExpand"
                                        TextColor="Black" />

                                    <Frame
                                        CornerRadius="20"
                                        Padding="0"
                                        IsClippedToBounds="True">

                                        <Image
                                            x:Name="image"
```

```
44                          Source="{Binding imageURL
                                }"
45                          HorizontalOptions="Center"
46                          VerticalOptions="Center"
                                ></Image>
47                  </Frame>
48
49                  <StackLayout
50                      Orientation="Horizontal" >
51
52                      <AbsoluteLayout
53                          AbsoluteLayout.LayoutFlags
                                ="All"
54                          AbsoluteLayout.
                                LayoutBounds ="40,
                                20,300, 40">
55
56                          <Label Margin="50,0,20,0
                                "
57                              Text="Tipo:"
58                              TextColor="Black"
                                    ></Label>
59
60                          <Label Margin="
                                120,0,20,0"
61                              Text="{Binding
                                    type}"
62                              TextColor="#2196F3"
                                    ></Label>
63
64                          <Label Margin="
                                220,0,20,0"
65                              Text="Difficolta:"
66                              TextColor="Black"
                                    ></Label>
67
68                          <Label Margin="
                                290,0,20,0"
69                              Text="{Binding
                                    difficulty}"
70                              TextColor="#2196F3"
                                     ></Label>
71
72                      </AbsoluteLayout>
73
74                  </StackLayout>
75
76                  <StackLayout
77                      Orientation="Horizontal" >
78
79                      <AbsoluteLayout
```

```
80                                          AbsoluteLayout.LayoutFlags
                                                ="All"
81                                          AbsoluteLayout.
                                                LayoutBounds ="40,
                                                20,300, 40">
82
83                                          <Label Margin="50,0,20,0
                                                "
84                                                  Text="Tempo:"
85                                                  TextColor="Black"
                                                        ></Label>
86
87                                          <Label Margin="
                                                120,0,20,0"
88                                                  Text="{Binding
                                                        time}"
89                                                  TextColor="#2196F3"
                                                        ></Label>
90
91                                          <Label Margin="
                                                220,0,20,0"
92                                                  Text="Prezzo:"
93                                                  TextColor="Black"
                                                        ></Label>
94
95                                          <Label Margin="
                                                290,0,20,0"
96                                                  Text="{Binding
                                                        price}"
97                                                  TextColor="#2196F3"
                                                        ></Label>
98                                      </AbsoluteLayout>
99                                  </StackLayout>
100
101                             </StackLayout>
102
103
104
105                 </ViewCell>
106                 </DataTemplate>
107
108             </ListView.ItemTemplate>
109         </ListView>
110
111     </StackLayout>
112 </ContentPage>
```

Listato 4.8: Implementazione grafica della Recipe Page

4.3.3 Login page e Register page

Il funzionamento della pagina di login è molto semplice. Infatti, l'utente inserisce la propria email e la password e, grazie al comando `LoginCommand`, situato nel Listato 4.11, tali dati vengono inviati al LoginViewModel (Listato 4.10) che comunica con il Model attraverso la funzione `LoginAsync(Email,Password)` (mostrato nel Listato 4.4). Se la risposta del server è positiva vuol dire che il login è andato a buon fine; altrimenti vuol dire che c'è stato qualche errore. Nella schermata del login l'utente ha anche la possibilità di registrarsi, grazie all'apposito pulsante che, come si può notare nel Listato 4.9, richiama la pagina della registrazione attraverso il metodo `Button_Clicked()`.

Implementazione della Login Page

```
1   using ApplicazioneEAToday.Models;
2   using System;
3   using System.Collections.Generic;
4   using System.Linq;
5   using System.Text;
6   using System.Threading;
7   using System.Threading.Tasks;
8
9   using Xamarin.Forms;
10  using Xamarin.Forms.Xaml;
11
12  namespace ApplicazioneEAToday.Views
13  {
14      [XamlCompilation(XamlCompilationOptions.Compile)]
15      public partial class LoginPage : ContentPage
16      {
17          public LoginPage()
18          {
19              InitializeComponent();
20              Title = "LOGIN";
21          }
22
23          private async void Button_Clicked(object sender, EventArgs e)
24          {
25              await Navigation.PushAsync(new RegisterPage());
26          }
27
28          protected override bool OnBackButtonPressed()
29          {
30              base.OnBackButtonPressed();
31              return true;
32          }
33
34
35      }
36  }
```

Listato 4.9: Implementazione della Login page

```
using ApplicazioneEAToday.Services;
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Input;
using Xamarin.Forms;
using ApplicazioneEAToday.Models;
using ApplicazioneEAToday.Views;

namespace ApplicazioneEAToday.ViewModels
{
    public class LoginViewModel
    {

        private ApiServices _apiServices = new ApiServices();
        public string Email { get; set; }
        public string Password { get; set; }
        public string Message { get; set; }

        public ICommand LoginCommand
        {
            get
            {
                return new Command(async() =>
                {
                    var isSuccess=await _apiServices.LoginAsync(Email,
                        Password);
                    if (isSuccess) {
                        User.log = true;
                        Message = "login eseguito";

                    }
                    else
                        Message = "Email o password sbagliati";
                    OpenNewPage();
                });
            }

        }
        private async void OpenNewPage()
        {

            if (User.log)
            {
                await App.Current.MainPage.DisplayAlert("Login
                    Success", "", "Ok");
                App.Current.MainPage = new MainPage();
```

```
46                  }
47                  else
48                  {
49                      await App.Current.MainPage.DisplayAlert("Login Fail",
50                                      "Inserisci correttamente Email e
                                            Password", "OK");
51
52                  }
53              }
54
55          }
56
57      }
```

Listato 4.10: Implementazione del LoginViewModel

Implementazione della grafica della Login page

```
1
2       <?xml version="1.0" encoding="utf-8" ?>
3   <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
4                xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5                xmlns:vm="clr-namespace:ApplicazioneEAToday.ViewModels"
6                x:Class="ApplicazioneEAToday.Views.LoginPage"
7                BackgroundColor="White">
8
9
10          <ContentPage.BindingContext>
11              <vm:LoginViewModel/>
12          </ContentPage.BindingContext>
13
14              <StackLayout Margin="25"
15                           VerticalOptions="Center"
16                           Orientation="Vertical">
17
18                  <Label Text="Inserisci la tua email: "
19                         TextColor="Black"
20                         FontSize="Subtitle"></Label>
21
22                  <Entry Text="{Binding Email}"
23                         Placeholder="Email"/>
24
25                  <Label Text="Inserisci la tua password: "
26                         TextColor="Black"
27                         FontSize="Subtitle"></Label>
28
29                  <Entry Text="{Binding Password} "
30                         Placeholder="Password"
31                         IsPassword="True"/>
32
33                  <Button Margin="80,0"
```

```
                    Command="{Binding LoginCommand}"
                    Text="Login"
                    TextColor="White"
                    BackgroundColor="#2196F3"/>

            <Label Text="Oppure"
                   TextColor="Black"
                   HorizontalOptions="CenterAndExpand"></Label>

            <Button Margin="80,0"
                    Text="Registrati"
                    TextColor="White"
                    BackgroundColor="#2196F3"
                    Clicked="Button_Clicked"></Button>

        </StackLayout>
</ContentPage>
```

Listato 4.11: Implementazione della grafica della Login page

Register page

Nella schermata della registrazione all'utente vengono richiesti i dati da inserire e, una volta inseriti tali dati, questi vengono inviati al `RegisterViewModel` (Listato 4.13) attraverso il comando `RegisterCommand` (Listato 4.12). Qui, dopo vari controlli sui dati inseriti, viene chiamato il metodo `RegisterAsync(Email, Name, LastName, Address, Password)`, mostrato nel Listato 4.4. Se la registrazione va a buon fine, l'utente, attraverso il metodo `OpenNewPage()`, viene indirizzato alla schermata di login, dove può effettuare il login.

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="ApplicazioneEAToday.Views.RegisterPage"
             xmlns:vm="clr-namespace:ApplicazioneEAToday.ViewModels"
             BackgroundColor="White">
    <ContentPage.BindingContext>
        <vm:RegisterViewModels/>
    </ContentPage.BindingContext>

    <StackLayout Margin="25"
                 Orientation="Vertical"
                 VerticalOptions="Center">

        <Label Text="Inserisci i tuoi dati:"
               TextColor="Black"
               FontSize="Title"></Label>

        <Entry Text="{Binding Name}"
               Placeholder="Nome"/>

```

```
23          <Entry Text="{Binding LastName}"
24                 Placeholder="Cognome"/>
25
26          <Entry Text="{Binding Address}"
27                 Placeholder="Indirizzo"/>
28
29          <Entry Text="{Binding Email}"
30                 Placeholder="Email"/>
31
32          <Entry Text="{Binding Password}"
33                 Placeholder="Password"
34                 IsPassword="True"/>
35
36          <Entry Text="{Binding ConfirmPassword}"
37                 Placeholder="Conferma password"
38                 IsPassword="True"/>
39
40          <Button Margin="70,30"
41                  TextColor="White"
42                  BackgroundColor="#2196F3"
43                  Command="{Binding RegisterCommand}"
44                  Text="Registrati"/>
45
46      </StackLayout>
47  </ContentPage>
```

Listato 4.12: Implementazione della grafica della Register page

```
1
2   using ApplicazioneEAToday.Services;
3   using ApplicazioneEAToday.Views;
4   using System;
5   using System.Collections.Generic;
6   using System.Text;
7   using System.Windows.Input;
8   using Xamarin.Forms;
9
10  namespace ApplicazioneEAToday.ViewModels
11  {
12      public class RegisterViewModels
13      {
14          ApiServices _apiServices = new ApiServices();
15
16          public string Email { get; set; }
17          public string Password { get; set; }
18          public string ConfirmPassword { get; set; }
19          public string Name { get; set; }
20          public string LastName { get; set; }
21          public string Address { get; set; }
22          public string Message { get; set;}
23
```

```
24          public ICommand RegisterCommand
25          {
26              get
27              {
28                  return new Command(async() =>
29                  {
30                      if (Email == null || Name == null || LastName ==
                            null ||
31                          Address == null || Password == null)
32                      {
33
34                          await App.Current.MainPage.DisplayAlert("Ops",
35                                  "Perfavore inserisci tutti i dati",
                                        "Riprova");
36                      }
37                      else
38                      {
39                          if (Password.Equals(ConfirmPassword) && !
                                Password.Equals(""))
40                          {
41                              var isSuccess = await _apiServices.
                                    RegisterAsync(Email,
42                                                  Name, LastName,
                                                        Address, Password);
43
44                              if (isSuccess)
45                              {
46                                  Message = "Registered successfully";
47                                  OpenNewPage();
48
49                              }
50                              else
51                                  Message = "Retry later";
52                          }
53                          else
54                          {
55                              await App.Current.MainPage.DisplayAlert("
                                    Ops",
56                                  "Perfavore inserisci la stessa
                                        password", "Riprova");
57
58                          }
59                      }
60                  });
61              }
62
63
64          }
65          private async void OpenNewPage()
66          {
67              await App.Current.MainPage.DisplayAlert("Ti sei
```

```
                    registrato con successo", "", "Ok");
68                  App.Current.MainPage = new LoginPage();
69
70              }
71          }
72      }
```

Listato 4.13: Implementazione del RegisterViewModel

4.3.4 Filtraggio delle ricette

Il filtraggio delle ricette è una funzionalità che fa si che l'utente può selezionare diversi alimenti da includere e escludere dalla ricerca, selezionare la difficoltà e il tipo di ricetta che vuole cercare. Nel Listato 4.14 si nota che il metodo `creazioneStringaURLfilter()` gioca un ruolo fondamentale perchè esso genera la stringa, in base ai dati inseriti dall'utente, che verrà utilizzata come URL nel metodo del Model `RecipeAsync()` (Listato 4.4); quest'ultimo farà vedere solo le ricette che rispettano la richiesta dell'utente (ovviamente la funzione che filtra le ricette è mostrato in back-end nel Listato 4.3).

Implementazione della funzionalità di filtraggio delle ricette

```
1   using ApplicazioneEAToday.Models;
2   using System;
3   using System.Collections.Generic;
4   using System.Collections.ObjectModel;
5   using System.Linq;
6   using System.Text;
7   using System.Threading.Tasks;
8
9   using Xamarin.Forms;
10  using Xamarin.Forms.Xaml;
11
12  namespace ApplicazioneEAToday.Views
13  {
14      [XamlCompilation(XamlCompilationOptions.Compile)]
15      public partial class CosaCucino : ContentPage
16      {
17          ObservableCollection<Ingredient> listaInclusi { get; set; }
18          ObservableCollection<Ingredient> listaEsclusi { get; set; }
19          public static string get = "";
20          public static ObservableCollection<Ingredient> listacompleta;
21
22          public CosaCucino ()
23          {
24              InitializeComponent ();
25              Title = "Cosa Cucino";
26              listacompleta= RecipePage.listaCompletaIngredienti;
```

```
27              listaInclusi  = new ObservableCollection<Ingredient>();
28              listaEsclusi  = new ObservableCollection<Ingredient>();
29              listaIngredientiInclusi .HeightRequest = 0;
30              listaIngredientiEsclusi .HeightRequest = 0;
31
32          }
33
34          public void Button_Clicked(object sender, EventArgs e)
35          {
36              string ingredienteIncluso = ingredienteincluso.Text;
37
38              for (int i =0; i <= listacompleta.Count; i++)
39              {
40                  if (listacompleta[i].name.Contains(ingredienteIncluso))
41                  {
42                      listaInclusi .Add(listacompleta[i]);
43
44                      ingredienteincluso.Text = "";
45                      ingredienteincluso.Placeholder=" Nome";
46                      listaIngredientiInclusi .HeightRequest = (
                            listaInclusi.Count+1)*20;
47                      break;
48
49                  }
50              }
51              listaIngredientiInclusi .ItemsSource = listaInclusi;
52
53          }
54          public void Button_Clicked1(object sender, EventArgs e)
55          {
56              string ingredienteEscluso = ingredienteescluso.Text;
57
58              for (int i = 0; i <= listacompleta.Count; i++)
59              {
60                  if (listacompleta[i].name.Contains(ingredienteEscluso))
61                  {
62                      listaEsclusi .Add(listacompleta[i]);
63
64                      ingredienteescluso.Text = "";
65                      ingredienteescluso.Placeholder = " Nome";
66                      listaIngredientiEsclusi .HeightRequest = (
                            listaEsclusi.Count + 1) * 20;
67                      break;
68                  }
69              }
70              listaIngredientiEsclusi .ItemsSource = listaEsclusi;
71
72          }
73
74          private void Button_Clicked_1(object sender, EventArgs e)
75          {
```

```
76                  if (button2.IsVisible)
77                  {
78                      button2.IsVisible = false;
79                  }
80                  else      button2.IsVisible = true;
81
82                  if (button3.IsVisible)
83                  {
84                      button3.IsVisible = false;
85                  }
86                  else button3.IsVisible = true;
87
88
89              }
90              private void Button_Clicked_2(object sender, EventArgs e)
91              {
92                  if (button1.IsVisible)
93                  {
94                      button1.IsVisible = false;
95                  }
96                  else button1.IsVisible = true;
97
98                  if (button3.IsVisible)
99                  {
100                     button3.IsVisible = false;
101                 }
102                 else button3.IsVisible = true;
103
104
105             }
106             private void Button_Clicked_3(object sender, EventArgs e)
107             {
108                 if (button2.IsVisible)
109                 {
110                     button2.IsVisible = false;
111                 }
112                 else button2.IsVisible = true;
113
114                 if (button1.IsVisible)
115                 {
116                     button1.IsVisible = false;
117                 }
118                 else button1.IsVisible = true;
119
120             }
121             public string creazioneStringaURLfilter()
122             {
123                  get = "?";
124                 if( listaInclusi .Count != 0)
125                 {
126                     get += "selected=";
```

```
127             int size = listaInclusi.Count() - 1;
128             for(int i = 0; i < size; i++)
129             {
130                 get += listaInclusi[i].id + ",";
131             }
132             get += listaInclusi[size].id;
133         }
134         if ( listaEsclusi.Count != 0)
135         {
136             if (!get.Equals("?"))
137             {
138                 get += "&";
139             }
140             get += "excluded=";
141             int size = listaEsclusi.Count() - 1;
142             for (int i = 0; i < size; i++)
143             {
144                 get += listaEsclusi[i].id + ",";
145             }
146             get += listaEsclusi[size].id;
147         }
148 
149         if (switch1.IsToggled)
150         {
151             if (!get.Equals("?"))
152             {
153                 get += "&";
154             }
155             get += "type=" + "vegetariano";
156         }
157         else
158         {
159             if (switch2.IsToggled)
160             {
161                 if (!get.Equals("?"))
162                 {
163                     get += "&";
164                 }
165                 get += "type=" + "vegano";
166             }
167         }
168         if (button1.IsVisible && button2.IsVisible && button2.
                IsVisible)
169         {
170             return get;
171         }else
172         {
173             if (!get.Equals("?"))
174             {
175                 get += "&";
176             }
```

```
177                     get += "difficulty=";
178                     if (button1.IsVisible)
179                         get += button1.Text;
180                     if (button2.IsVisible)
181                         get += button2.Text;
182                     if (button3.IsVisible)
183                         get += button3.Text;
184                 }
185                 return get;
186             }
187
188             private async void Button_Click(object sender, EventArgs e)
189             {
190                 string URL = creazioneStringaURLfilter();
191                 string ingredienteincluso = URL;
192                 await Navigation.PushAsync(new RecipePage());
193             }
194
195             private void switch1_Toggled(object sender, ToggledEventArgs e)
196             {
197                 if (switch1.IsToggled)
198                 {
199                     switch2.IsToggled = true;
200                 }
201                 else switch2.IsToggled = false;
202             }
203
204
205
206             private void listaIngredientiEsclusi_ItemTapped(object sender,
                    ItemTappedEventArgs e)
207             {
208                 Ingredient remove = e.Item as Ingredient;
209                 for (int i = 0; i < listaEsclusi.Count; i++)
210                 {
211                     if (listaEsclusi[i].name.Contains(remove.name))
212                     {
213                         listaEsclusi.Remove(listaEsclusi[i]);
214                         listaIngredientiEsclusi.HeightRequest = (
                                listaEsclusi.Count+1) * 20;
215                         break;
216                     }
217                 }
218             }
219
220             private void listaIngredientiInclusi_ItemTapped(object sender,
                    ItemTappedEventArgs e)
221             {
222                 Ingredient remove = e.Item as Ingredient;
223                 for (int i = 0; i < listaInclusi.Count; i++)
224                 {
```

```
225                     if ( listaInclusi [i].name.Contains(remove.name))
226                     {
227                         listaInclusi .Remove(listaInclusi[i]);
228                         listaIngredientiInclusi .HeightRequest = (
                                listaInclusi.Count +1 ) * 20;
229                         break;
230                     }
231                 }
232             }
233         }
234     }
```

Listato 4.14: Implementazione della funzionalità di filtraggio delle ricette

Implementazione grafica schermata Filtra Ricette

```
1
2     <?xml version="1.0" encoding="utf-8" ?>
3     <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
4                  xmlns:x="http://schemas.microsoft.com/winfx/2009/
                       xaml"
5                  x:Class="ApplicazioneEAToday.Views.CosaCucino"
6                  BackgroundColor="White">
7       <ScrollView>
8         <StackLayout HorizontalOptions="Center">
9             <StackLayout HorizontalOptions="Center">
10                <Label Margin="20,50,70,0"
11                       FontSize="Subtitle"
12                       TextColor="Black"
13                       Text="Ingredienti che vuoi includere nella
                              ricetta:"></Label>
14
15                <StackLayout HorizontalOptions="Center"
16                             Orientation="Horizontal">
17                    <Entry x:Name="ingredienteincluso"
18                           WidthRequest="200"
19                           Margin="20,0,30,0"
20                           Placeholder="Nome"/>
21                    <Button Text="Aggiungi"
22                            BackgroundColor="White"
23                            TextColor="#2196F3"
24                            Clicked="Button_Clicked"></Button>
25                </StackLayout>
26
27                    <ListView Margin="0,0,0,0"
28                              x:Name="listaIngredientiInclusi"
29                              ItemTapped="
                                    listaIngredientiInclusi_ItemTapped"
30                              HasUnevenRows="true" >
31                        <ListView.ItemTemplate>
32                            <DataTemplate>
```

```
33                              <ViewCell >
34                                  <Label FontSize="Subtitle"
35                                     TextColor="Gray"
36                                     Text="{Binding name}"
37                                     Margin="55,0,0,0" />
38                              </ViewCell>
39                          </DataTemplate>
40                      </ListView.ItemTemplate>
41                  </ListView>
42
43              <Label Margin="20,0,70,0"
44                     FontSize="Subtitle"
45                     TextColor="Black"
46                     Text="Ingredienti che vuoi escludere nella
                           ricetta:"></Label>
47
48              <StackLayout HorizontalOptions="Center"
49                           Orientation="Horizontal">
50                  <Entry x:Name="ingredienteescluso"
51                         WidthRequest="200"
52                         Margin="20,0,30,0"
53                         Placeholder="Nome"/>
54                  <Button Text="Aggiungi"
55                          BackgroundColor="White"
56                          TextColor="#2196F3"
57                          Clicked="Button_Clicked1" ></Button>
58              </StackLayout>
59
60                  <ListView Margin="0,0,0,0"
61                            x:Name="listaIngredientiEsclusi"
62                            ItemTapped="
                                 listaIngredientiEsclusi_ItemTapped"
63                            HasUnevenRows="true" >
64                      <ListView.ItemTemplate>
65                          <DataTemplate >
66                              <ViewCell >
67                                      <Label TextColor="Gray"
68                                             FontSize="Subtitle"
69                                             Text="{Binding name}"
70                                             Margin="55,0,0,0"/>
71                              </ViewCell>
72                          </DataTemplate>
73                      </ListView.ItemTemplate>
74                  </ListView>
75
76              <Label Margin="30,0,0,0"
77                     TextColor="Black"
78                     Text="Scegli scegli il tipo di ricetta:"
79                     FontSize="Subtitle"></Label>
80
81              <StackLayout Margin=" 0,0,0,0"
```

```
                Orientation="Horizontal">
    <Label Margin="50,0,0,0"
           Text="Vegetariano"
           FontSize="Subtitle"></Label>
    <Switch Margin="150,0,0,0"
            x:Name="switch1"
            Toggled="switch1_Toggled"
            OnColor="#2196F3"></Switch>
</StackLayout>

<StackLayout Margin="0,0,0,0"
             Orientation="Horizontal">
    <Label Margin="50,0,0,0"
           Text="Vegano"
           FontSize="Subtitle"></Label>
        <Switch Margin="180,0,0,0"
                x:Name="switch2"
                OnColor="#2196F3"></Switch>
</StackLayout>

<Label Margin="30,0,0,0"
       TextColor="Black"
       Text="Scegli la difficolta della ricetta:"
       FontSize="Subtitle"></Label>

<StackLayout HorizontalOptions="Center"
             Margin="40,30,30,0"
             Orientation="Horizontal">
        <Button Margin="0,0,0,0"
                BackgroundColor="Azure"
                x:Name="button1"
                TextColor="#2196F3"
                Text="facile"
                Clicked="Button_Clicked_1"></Button>
        <Button Margin="10,0,0,0"
                BackgroundColor="Azure"
                x:Name="button2"
                TextColor="#2196F3"
                Text="media"
                Clicked="Button_Clicked_2"></Button>
        <Button Margin="10,0,0,0"
                BackgroundColor="Azure"
                x:Name="button3"
                TextColor="#2196F3"
                Text="difficile"
                Clicked="Button_Clicked_3"></Button>
</StackLayout>

    <Button Margin="80,30,80,50"
            HorizontalOptions="Center"
            Text="Filtra Ricette"
```

```
                        TextColor="White"
                        BackgroundColor="#2196F3"
                        Clicked="Button_Click"></Button>

            </StackLayout>
        </StackLayout>

    </ScrollView>

</ContentPage>
```

Listato 4.15: Implementazione della grafica di filtraggio delle ricette

4.3.5 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*: consiste in una FAQ che l'utente può consultare per scoprire qualcosa in più sui creatori del progetto e sullo scopo dell'applicazione.
- *Schermata profilo*: l'utente può accedere a questa activity soltanto se ha già effettuato il login. 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.

4.4 Manuale utente

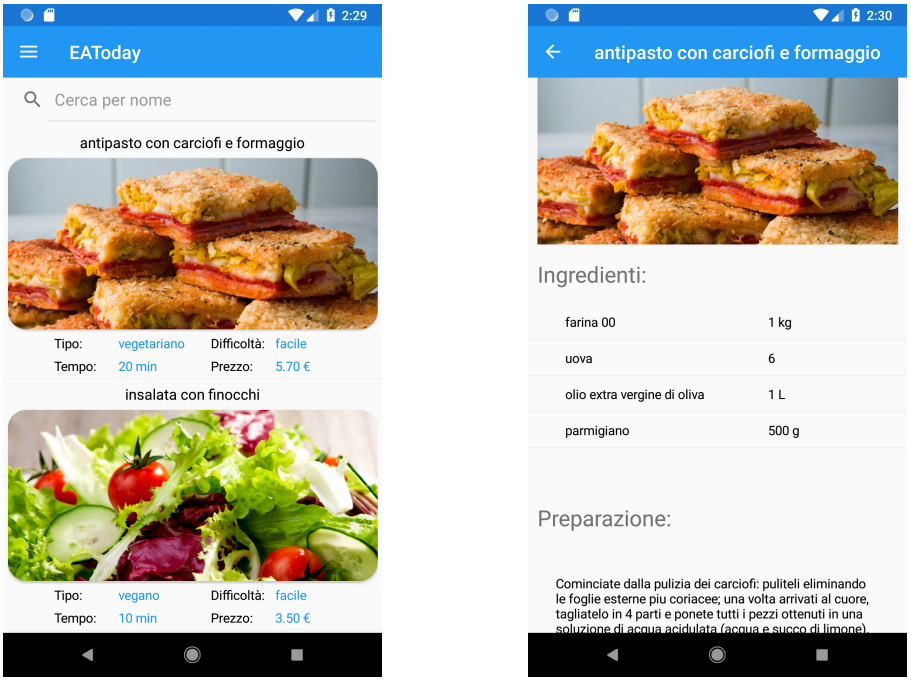
4.4.1 Schermata iniziale dell'app

La home page di EAToday (Figura 4.2(a)) coincide con la schermata che mostra le varie ricette. All'apertura dell'app, quindi, l'utente può iniziare a scorrere la home page per scoprire le ricette disponibili; successivamente egli potrà premere su una di queste 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 la ricetta è adatta a vegetariani e/o vegani; tutte queste informazioni risultano estremamente utili per la scelta della ricetta da preparare. In alternativa, l'utente può utilizzare la barra di ricerca, posta in alto 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.

4.4.2 Schermata dettagli ricetta

Quando l'utente seleziona una ricetta, la schermata è quella mostrata in Figura 4.2(b), dove vengono mostrati gli ingredienti di una ricetta, la sua preparazione e si ha la possibilità di ordinare gli ingredienti stessi. Se l'utente seleziona la voce "Ordina" e non è registrato viene invitato ad effettuare il login (o la registrazione).

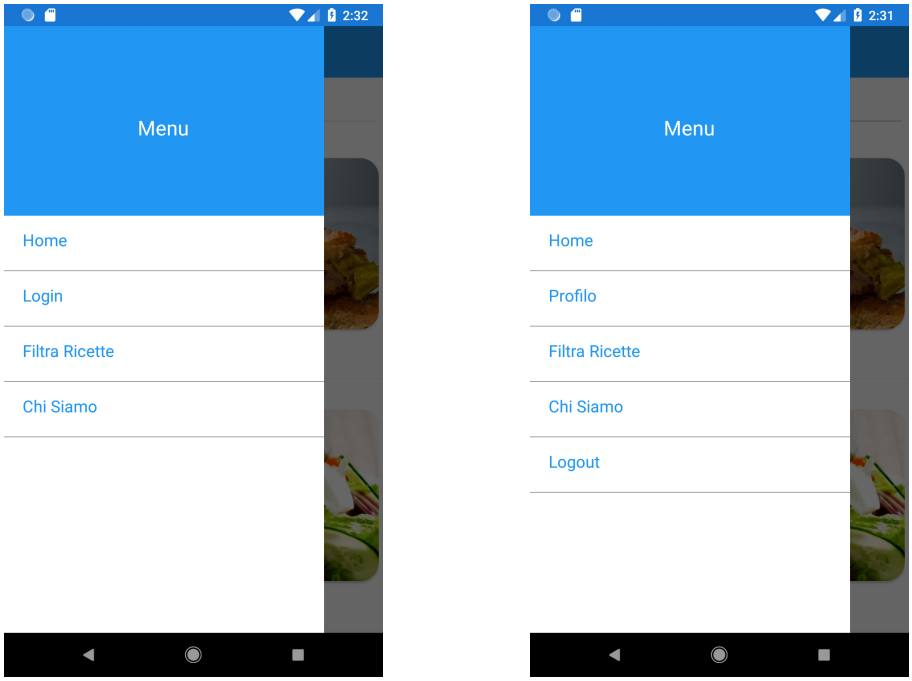


(a) *"Home page "* (b) *"Dettagli ricetta"*

Figura 4.2: Schermata principale e schermata di dettaglio di una ricetta

4.4.3 Schermata del menù

La schermata del menù è molto semplice; essa però, come si può notare nelle Figure 4.3(a) e 4.3(b), cambia in base al fatto che l'utente abbia effettuato il login o meno. Se l'utente non si è connesso,potrà accedere alla voce "Login" del menù per effettuare l'accesso o registrarsi. Se è connesso, nel menù, al posto della voce "Login", sono poste la voce "Profilo", in cui vengono mostrati i dati dell'utente, e la voce "Logout", con la quale l'utente si può disconnettere. L'utente può selezionare le varie opzioni in base a quello che desidera fare.



(a) *"Menù"* (b) *"Menù con login effettuato"*

Figura 4.3: Schermata del menù con i due casi possibili

4.4.4 Schermata di login

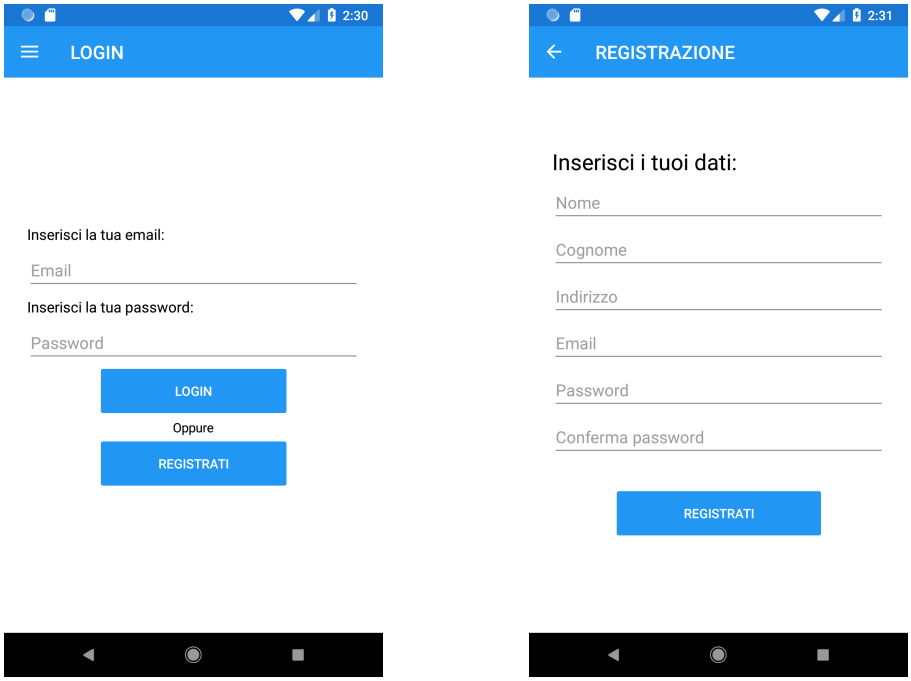
In Figura 4.4(a) viene rappresentata la schermata che si ottiene quando viene selezionata la voce "Login" nel menù. L'utente deve inserire i propri dati e premere il pulsante di login. Se non è registrato può effettuare la registrazione attraverso l'apposito pulsante "Registrati".

4.4.5 Schermata di registrazione

Quando l'utente ha intenzione di registrarsi a partire dalla schermata login, la schermata della registrazione è quella in Figura 4.4(b). In tale schermata l'utente deve inserire i propri dati e premere l'apposito pulsante.

4.4.6 Schermata di filtraggio delle ricette

Quando l'utente seleziona la voce "Filtra ricette", si apre la schermata in Figura 4.5(a) dove egli può selezionare gli ingredienti da includere e/o escludere; ciò avviene attraverso il pulsante "Aggiungi". L'utente può, oltre, selezionare il tipo di ricetta attraverso i due *switch*, e la difficoltà attraverso il pulsante corrispondente.



(a) *"Schermata del Login"* (b) *"Schermata di Registrazione"*

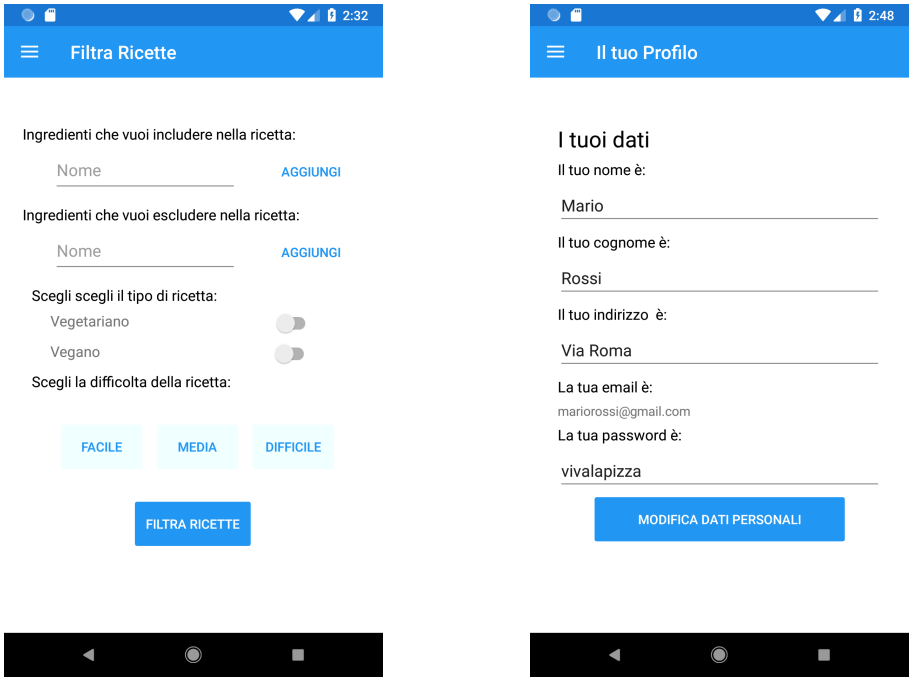
Figura 4.4: Schermata di login e schermata relativa alla registrazione

4.4.7 Schermata del profilo

Affinchè l'utente possa accedere alla schermata del "Profilo" (Figura 4.5(b)), egli deve avere precedentemente effettuato il login. In tale schermata l'utente può visualizzare i propri dati.

4.4.8 Schermata del progetto

La schermata "Progetto" permette all'utente di scoprire qualcosa in più sui creatori del progetto e sullo scopo dell'applicazione.



(a) *"Schermata Filtra ricette"* (b) *"Schermata del profilo utente"*

Figura 4.5: Schermata di filtraggio delle ricette e schermata relativa al profilo

5

Discussione, conclusioni e sfide future

Questo capitolo inizia con una breve discussione dell'applicazione; successivamente, esso procede con l'analisi dei punti di forza e di debolezza dell'App EAToday, e con l'esame dei possibili miglioramenti futuri

5.1 Discussione

EAToday è un'applicazione che è stata pensata per tentare di combattere lo spreco alimentare. Tutti quei prodotti che hanno una scadenza a breve termine rischiano di essere gettati. Infatti, circa un terzo del cibo prodotto sul nostro pianeta viene sprecato ancora prima di arrivare a tavola, e questo fatto ci fa pensare a quanto cibo viene sprecato inutilmente. Questo dato è estremamente allarmante ed è importante fare qualcosa per tentare di migliorare questa situazione, anche agendo nel piccolo del quotidiano. Quest'App si pone nell'ottica di dare un piccolo contributo nel combattere tale spreco. Lo scopo è, infatti, quello di permettere all'utente di acquistare prodotti alimentari con data molto vicina a quella della scadenza. L'App mostra delle ricette, che tengono conto degli alimenti in scadenza, disponibili nei vari negozi, e informa l'utente della difficoltà e del tempo di preparazione della ricetta selezionata. Con una semplice applicazione si vuole combattere in modo molto immediato un problema sempre più presente al giorno d'oggi. Come ogni prodotto, anche l'applicazione EAtoday ha dei *punti di forza* e dei *punti di debolezza*, che analizzeremo nelle prossime sottosezioni:

5.1.1 Punti di forza

In questa sezione vediamo tutto ciò che può essere considerato un vantaggio dell'applicazione EAToday. Un punto di forza relativo all'App realizzata è, sicuramente, rappresentato dalla modalità di sviluppo dell'applicazione. Infatti, è stato utilizzato il framework Xamarin; esso abbatte gli svantaggi delle altre tipologie di sviluppo e, rispetto allo sviluppo nativo, si riducono i costi, i tempi di sviluppo e la complessità di manutenzione dell'applicazione. Un ulteriore punto di forza è rappresentato dal

fatto che almeno il 60% del codice è condivisibile tra le varie piattaforme; è questo che permette di ridurre i costi e i tempi di sviluppo quando si vuole realizzare un'App per tutti e tre i principali store (Android, IOS e Windows Phone).

Un altro vantaggio è, proprio, lo scopo per cui è stata realizzata l'applicazione, ovvero quello di diminuire lo spreco del cibo. Ci sono poche applicazioni che sono realizzate a tale scopo, un esempio è "Too Good To Go", ma l'idea è molto diversa da quella del'App EAToday. Un altro punto a favore è proprio il fatto che non c'è tanta concorrenza, quindi è molto facile fare pubblicità e fare conoscere l'applicazione in giro.

L'applicazione è molto semplice da utilizzare e, grazie a ciò, può essere adottata anche da chi ha scarsa dimestichezza con i dispositivi mobili.

L'applicazione può aiutare l'utente meno esperto a cucinare, grazie alla schermata degli ingredienti che contiene, anche, il procedimento della ricetta.

Grazie all'applicazione EAToday, anche i negozi stessi avranno più guadagno poiché getteranno meno alimenti.

5.1.2 Punti di debolezza

Il principale problema dell'App EAToday è il fatto che l'applicazione ha sempre bisogno che il dispositivo sia connesso a Internet, altrimenti l'utente può solo visualizzare le ricette che ha salvato.

Un altro problema è la sicurezza informatica; infatti visto che salviamo i dati dell'utente e delle ricette in un database online, questo può essere una preda facile di una SQL injection, ovvero una tecnica di code injection, usata per attaccare applicazioni di gestione di dati. Tale tecnica permette agli attaccanti di effettuare *spoof identify*, ovvero di modificare dati esistenti, di causare l'annullamento di transazioni o la modifica dei bilanci, nonchè di ottenere tutti i dati del sistema, di eliminare dati oppure fare in modo che non siano accessibili. La stessa tecnica di attacco consente, altrisì, di diventare amministratori del database server.

5.1.3 Lesson Learned

La lezione appresa più importante è proprio l'utilizzo del framework "Xamarin" per lo sviluppo di un'applicazione mobile. Durante la realizzazione dell'applicazione EAToday siamo riusciti a capire come utilizzare al meglio la documentazione del linguaggio di programmazione C#, ma soprattutto a comprendere eventuali errori grazie all'utilizzo del debugger. Molto importante è il fatto di essere ordinati durante la scrittura del codice perché, in questo modo, è sia più facile ricontrollare eventuali errori sia rendere il codice più comprensibile da parte di programmatori terzi.

5.2 Conclusioni e sfide future

Per la realizzazione del progetto nella presente tesi è stata necessaria una prima parte dedicata all'analisi dei requisiti in cui sono state individuate le funzionalità che dovevano essere implementate nell'App. Successivamente, si è passati alla progettazione del database e dell'applicazione.

Una volta terminata la progettazione, si è passati alla parte dell'implementazione in cui è stato spiegato il back-end anche attraverso l'illustrazione e la spiegazione di alcuni listati. Successivamente si è passati all'illustrazione del front- end, spiegando le schermate principali dell'applicazione e come sono state implementate, attraverso la descrizione di parti di codice. Infine, è stato realizzato un manuale utente per aiutare quest'ultimo all'utilizzo dell'App, e si è visto quali sono i punti di forza e di debolezza dell'applicazione.

Nel futuro pensiamo di aggiungere all'App EAToday il servizio di geolocalizzazione, in modo da poter mostrare all'utente solo le ricette con gli ingredienti disponibili nei negozi all'interno di un determinato raggio, specificato dall'utente stesso. L'applicazione si può migliorare dal punto di vista della sicurezza aggiungendo dei token per quanto riguarda il login, ed aumentando la privacy sui dati dell'utente.

Riferimenti bibliografici

1. App native, ibride e Web App. `http://www.mr-apps.com/it/blog/la-differenza-tra-app-native-app-ibride-e-web-app`, 2019.
2. App native, ibride e Web App. `https://www.html.it/faq/app-mobile-ibride-native-o-web-le-differenze/`, 2019.
3. Guida PHP. `https://www.html.it/guide/guida-php-di-base/`, 2019.
4. Guida XML. `https://www.html.it/guide/guida-xml-di-base`, 2019.
5. Linguaggio di programmazione C#. `https://it.wikipedia.org/wiki/C_sharp`, 2019.
6. Platform Architecture. `https://developer.android.com/guide/platform/`, 2019.
7. Smartphone. `https://it.wikipedia.org/wiki/Smartphone`, 2019.
8. Xamarin. `https://visualstudio.microsoft.com/it/xamarin/`, 2019.
9. Xamarin.Forms. `https://docs.microsoft.com/it-it/xamarin/xamarin-forms/`, 2019.
10. C. Bilgin. *Mastering Cross-Platform Development with Xamarin*. Packt Publishing, 2016.
11. D. Bochicchio and C. Civera. *C# e Visual Studio 2015: Guida Completa per lo sviluppatore*. Hoepli, 2015.
12. M. Canducci. *XML. Conoscere il linguaggio XML significa poter comunicare veramente con tutti*. Apogeo, 2009.
13. E. Cisotti and M. Giannino. *Android: Guida Completa*. Edizioni LSWR, 2015.
14. A. Fuggetta, A. Binato, and L. Sfardini. *Ingegneria del software*. Creativi Pearson Education, 2006.
15. D. Hermes. *Xamarin Mobile Application Development*. Apress, 2015.
16. M. Jazayeri, C. Ghezzi, and D. Mandrioli. *Ingegneria del software. Fondamenti e principi*. Pearson Education, 2004.
17. C. Larman. *Applicare UML e i pattern. Analisi e progettazione orientata agli oggetti*. 2016.
18. M. Leibowitz. *Xamarin Mobile Development for Android Cookbook*. Packt Publishing, 2015.
19. C. Nagel. *C# and the .NET Core 1.0*. Wrox Pr Inc, 2016.
20. J. Peppers. *Xamarin Crossplatform Application Development*. Packt Publishing, 2014.
21. R.S. Pressman. *Principi di Ingegneria del Software*. The McGraw-Hill Companies, 2008.
22. M. Reynolds. *Xamarin Essentials*. Packt Publishing, 2013.
23. M.C. Reynolds. *Xamarin Mobile Application Development for Android*. Packt Publishing, 2014.
24. I. Sommerville. *Ingegneria del software*. Pearson Education, 2007.
25. A. Troelsen and P. Japikse. *C# and the .NET 4.6 Framework*. Apress, 2015.

Ringraziamenti

La strada da percorrere non è stata semplice, ma ho imparato molto dalle esperienze vissute in questi tre anni.

Il più grande ringraziamento va ai miei famigliari, che mi hanno fornito sostentamento sia economico che morale nonostante le difficoltà e gli inconvenienti che in questi anni ci siamo ritrovati ad affrontare.

Ringrazio il gruppo di studio: Alkis, Albo, Gatto, Mighelì, Millo, Lollo, KC e Chiara, che in qualsiasi momento di bisogno mi sono stati sempre a fianco. Un particolare ringraziamento va a Francesco (Albo), coinquilino, compagno di corso e anche compagno di squadra, che nei momenti più difficili, quando ero quasi sul punto di mollare, mi ha sempre incoraggiato ed aiutato a superare qualsiasi difficoltà.

Un altro ringraziamento va a Davide, che, grazie alle sue conoscenze motorie, ha fatto in modo che durante questi anni fossi rimasto sempre in forma.

Un sentito grazie va agli amici di Furci, che mi hanno sempre accolto con gioia nelle occasioni in cui tornavo in patria.

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 Xamarin e per avermi fornito le conoscenze e delucidazioni per arrivare alla fine di questo percorso.