



Kulinaria

Android Framework

Indri Muska

A.A.: 2011/2012

Matricola: 426534

E-mail: indrimuska@gmail.com



**Scuola Superiore
Sant'Anna**
di Studi Universitari e di Perfezionamento

SOMMARIO

| | |
|---|-----------|
| INTRODUZIONE | 1 |
| COS'È KULINARIA..... | 1 |
| COME FUNZIONA | 1 |
| COMPATIBILITÀ E SVILUPPO..... | 1 |
| | |
| ASPETTO INIZIALE | 2 |
| | |
| MAINACTIVITY | 3 |
| LAYOUT | 3 |
| IMPLEMENTAZIONE | 3 |
| SliderAdapter | 4 |
| | |
| INVENTORYPAGE | 5 |
| ASPETTO | 5 |
| 1. <i>Lista degli ingredienti</i> | 5 |
| 2. <i>Aggiunta di un ingrediente</i> | 6 |
| 3. <i>Opzioni associate ad ogni ingrediente</i> | 6 |
| 4. <i>Modifica di un ingrediente</i> | 6 |
| 5. <i>Cancellazione di un ingrediente</i> | 6 |
| SHOWINGREDIENTDIALOG..... | 7 |
| 1. <i>Nuovo ingrediente</i> | 7 |
| 2. <i>Modifica ingrediente</i> | 9 |
| REFRESH | 9 |
| | |
| RECIPESPAGE | 10 |
| 1. <i>Lista delle portate</i> | 10 |
| 2. <i>Ricerca ricetta</i> | 11 |
| 3. <i>Risultati della ricerca</i> | 11 |
| 4. <i>Reset dei risultati della ricerca</i> | 12 |
| | |
| MENUPAGE | 13 |
| 1. <i>Header principale</i> | 13 |
| 2. <i>Aggiunta di una pietanza</i> | 13 |

| | |
|---|-----------|
| 3. Cambiamento del giorno di riferimento | 14 |
| 4. Opzioni associate ad ogni pasto | 15 |
| 5. Consumazione di un pasto..... | 15 |
| 6. Cancellazione di un pasto | 15 |
| SHOPPINGLISTPAGE | 16 |
| ASPETTO | 16 |
| 1. Lista dei prodotti..... | 17 |
| 2. Organizzazione dei prodotti | 17 |
| 3. Informazioni associate ad ogni prodotto..... | 17 |
| 4. Riepilogo delle informazioni | 18 |
| GROUPEDLISTADAPTER..... | 18 |
| RECIPESLISTACTIVITY | 20 |
| 1. Intestazione | 20 |
| 2. Elenco delle ricette..... | 21 |
| 3. Back button | 21 |
| RECIPEACTIVITY | 22 |
| 1. Top bar | 22 |
| 2. Descrizione della ricetta | 22 |
| DATABASEINTERFACE | 23 |
| IL DATABASE | 23 |
| Gli schemi..... | 23 |
| Inventory..... | 23 |
| Recipes..... | 23 |
| RecipesIngredients..... | 24 |
| Menu..... | 24 |
| Riferimenti agli schemi | 24 |
| DBHELPER | 25 |
| STRUTTURA DELL'APPLICAZIONE | 27 |
| ALTRO | 28 |
| DEVELOPING..... | 28 |
| REPOSITORY | 28 |
| IMPLEMENTAZIONI FUTURE | 28 |
| Multilingua | 28 |

| | |
|---|----|
| <i>Aggiunta/modifica/eliminazione ricette</i> | 29 |
| <i>Notifica delle scadenze</i> | 29 |
| <i>Notifica pasto</i> | 29 |
| <i>Integrazione calendario</i> | 29 |
| <i>Prezzi</i> | 29 |
| <i>Dieta</i> | 30 |

INTRODUZIONE

COS'È KULINARIA

Kulinaria è un'applicazione ad uso quotidiano che nasce dall'esigenza di poter scegliere ogni giorno cosa mangiare e/o bere senza doversi preoccupare del resto. Chiunque ne faccia uso si concentra solo sui menù che sceglie giorno per giorno o progetta i giorni precedenti, affidando tutto il compito della gestione dei prodotti necessari alla realizzazione e al completamento del pasto all'applicativo.

COME FUNZIONA

Per sapere quali sono gli ingredienti e i prodotti che servono per completare un pasto occorre innanzitutto conoscere due aspetti fondamentali:

- Le ricette delle portate che si vogliono inserire in menù, soprattutto per quanto concerne il numero di ingredienti e la quantità richiesta;
- L'inventario dei prodotti che ogni utente già possiede nella dispensa, nel frigo, o più in generale in cucina.

Sapendo quali prodotti sono richiesti per ogni ricetta e quali invece sono già in possesso dell'utente, è possibile definire una lista di *prodotti mancanti* che l'utente dovrà ottenere per poter concretizzare le scelte alimentari desiderate.

In questo modo si possono programmare colazioni, pranzi, cene o affini senza dover controllare o tenere a mente ogni volta quali prodotti sono già disponibili e quali no. Così facendo si unisce la comodità di un ricettario potatile con l'utilità di inventario personale, al fine di ottenere una lista della spesa intelligente, che si aggiorna automaticamente al variare delle proprie esigenze.

COMPATIBILITÀ E SVILUPPO

Nonostante l'aspetto grafico sia molto simile alle versioni più recenti di Android, l'applicazione è stata sviluppata per le versioni a partire dalla release 2.2 (Froyo), con le API di livello 8. Molte icone, colori o stili sono infatti stati realizzati seguendo le pratiche, i principi e i patterns descritti in *Android Design*¹, anch'esso aggiornato allo stile di Android 4, Gingerbread.

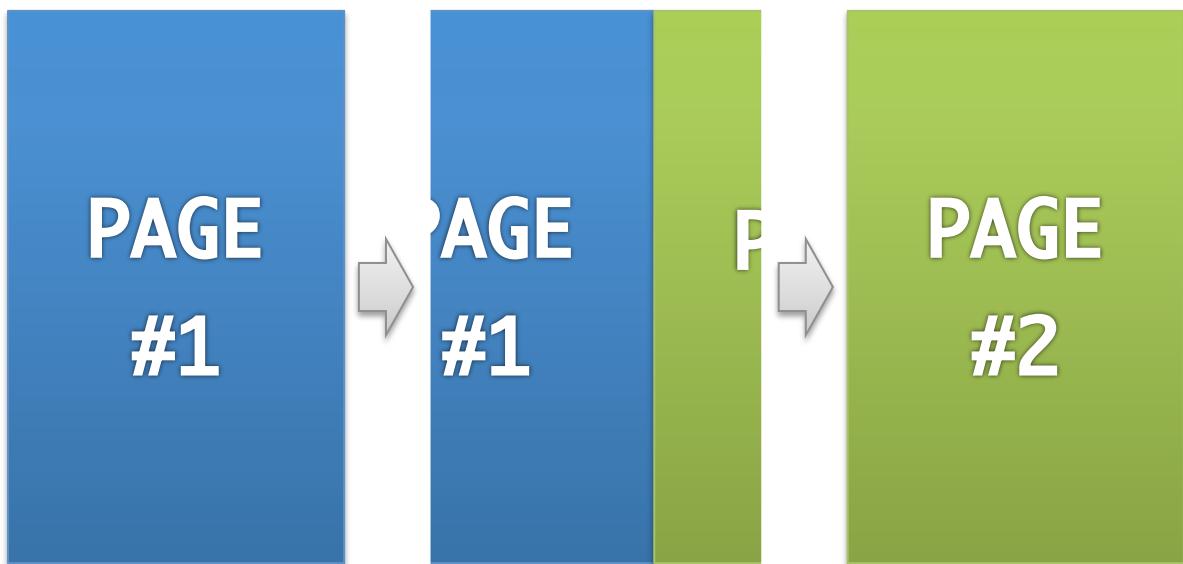
Per lo sviluppo si è scelto l'uso del software Eclipse in combinazione al relativo ADT plugin di Google.

¹ <http://developer.android.com/design/index.html>

ASPETTO INIZIALE

Prendendo spunto da alcune applicazioni abitualmente utilizzate e da altre presenti su *Google Play Store*, il primo aspetto che si è considerato nella definizione di *Kulinaria* è stato la struttura del layout dell'activity principale. Essendo il primo punto d'accesso all'applicazione, si è cercato di dargli un design che fosse il più possibile semplice, usabile e allo stesso tempo accattivante, in modo che gli utenti non si stanchino alla vista ogniqualvolta effettuano l'accesso.

Così facendo, si è scelto di utilizzare la vista *ViewPager* presente nel pacchetto di supporto agli sviluppatori², grazie alla quale è possibile scorrere lateralmente a destra e a sinistra tra pagine di dati senza cambiare activity, come mostrato nella figura che segue.



Di per sé, *ViewPager* fornisce soltanto la vista delle pagine, senza un indicatore globale che specifichi quante pagine sono presenti nello slider, e in quale pagina si sta visualizzando in questo momento. Per evitare di scrivere ex-novo un adattatore che fornisca tale supporto grafico ad ogni pagina, si è scelto di sfruttare un widget esterno al framework chiamato *ViewPagerIndicator*³, un componente molto semplice da usare che presenta già cinque diversi stili per la definizione delle interfacce degli indicatori di pagina.

Il tema usato per *Kulinaria* è *Tab Indicator* che mostra le etichette come un semplice *Tab Widget* con un indicatore sotto il testo che scorre con l'avanzare delle pagine. Per l'implementazione dell'adattatore allo slider e le relative caratteristiche si rimanda a [SliderAdapter](#).

² Il package `com.android.support.v4` non è incluso nel framework, ma è scaricabile dall'SDK Manager e va incluso come libreria nel progetto, così come indicato nel sito [Android Developers](#).

³ Documentazione e download sono reperibili all'indirizzo <http://viewpagerindicator.com/>.

MAINACTIVITY

MainActivity è l'activity principale che compare all'avvio dell'applicazione, perciò è registrata nel manifest all'intent LAUNCHER. Come tutte le activity dell'applicazione, si avvia solamente in modalità Portrait.

LAYOUT

Il layout è relativamente semplice perché presenta solamente una TextView in alto con il nome dell'applicazione su sfondo sfumato di blu e uno slider di pagine preceduto dal relativo indicatore.

IMPLEMENTAZIONE

Il compito principale della classe MainActivity è quello di gestire lo stato di ogni singola pagina contenuta al suo interno per permetterne l'interazione e l'aggiornamento nel modo più corretto. Tra gli attributi privati della classe c'è anche un oggetto di tipo DatabaseInterface che si pone a metà tra le activity e il database semplificando l'accesso e l'uso da parte di qualunque classe o metodo (vedi [DatabaseInterface](#)).

MainActivity estende la classe FragmentActivity e, in versione compressa, si presenta come segue:

```
public class MainActivity extends FragmentActivity {
    private static DatabaseInterface db;
    private ViewPager pager;

    // These pages are updated by other pages
    InventoryPage inventoryPage;
    ShoppingListPage shoppingListPage;

    @Override
    protected void onCreate(Bundle savedInstanceState) { ... }

    @Override
    protected void onPause() { ... }

    @Override
    protected void onResume() { ... }

    class SliderAdapter extends FragmentPagerAdapter implements TitleProvider { ...

        // Generic page to implement
        abstract class Page { ...

            // Slider's pages implementation
            final class MenuPage extends Page { ... }
            final class InventoryPage extends Page { ... }
            final class RecipesPage extends Page { ... }
            final class ShoppingListPage extends Page { ... }

            // Open the ingredient dialog
            public void showIngredientDialog(final String updateIngredientName) { ... }
        }
    }
}
```

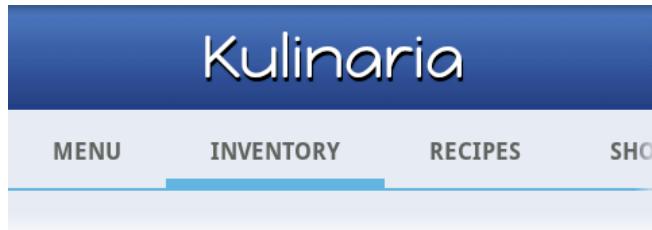
SLIDERADAPTER

SliderAdapter è una sottoclasse di MainActivity che implementa l'adattatore dell'oggetto ViewPager. In esso si rappresenta ogni pagina attraverso degli SliderFragment, creati attraverso il riferimento ad un vettore di oggetti derivati da Page e memorizzati come attributo privato:

```
// Generic page to implement
abstract class Page {
    protected String pageName;
    public Page(int name) { pageName = getString(name); }
    public String getTitle() { return pageName; }
    public abstract View getView();
    public abstract void refresh();
}
```

Ogni pagina dello slider estende Page, con l'evidente vantaggio di presentare obbligatoriamente metodi comuni (alcuni implementati, altri no) che posso essere richiamati sia da SliderFragment che da SliderAdapter. SliderFragment è l'estensione di un generico Fragment usato nell'adattatore dello slider, nel quale viene richiamato il metodo Page.getView() per creare la vista associata ad ogni pagina.

Ovviamente l'adattatore è legato all'indicatore delle pagine importato dal package com.viewpagerindicator, infatti implementa la classe TitleProvider che richiede l'override del metodo getTitle() per ottenere l'indicatore mostrato in figura seguente.

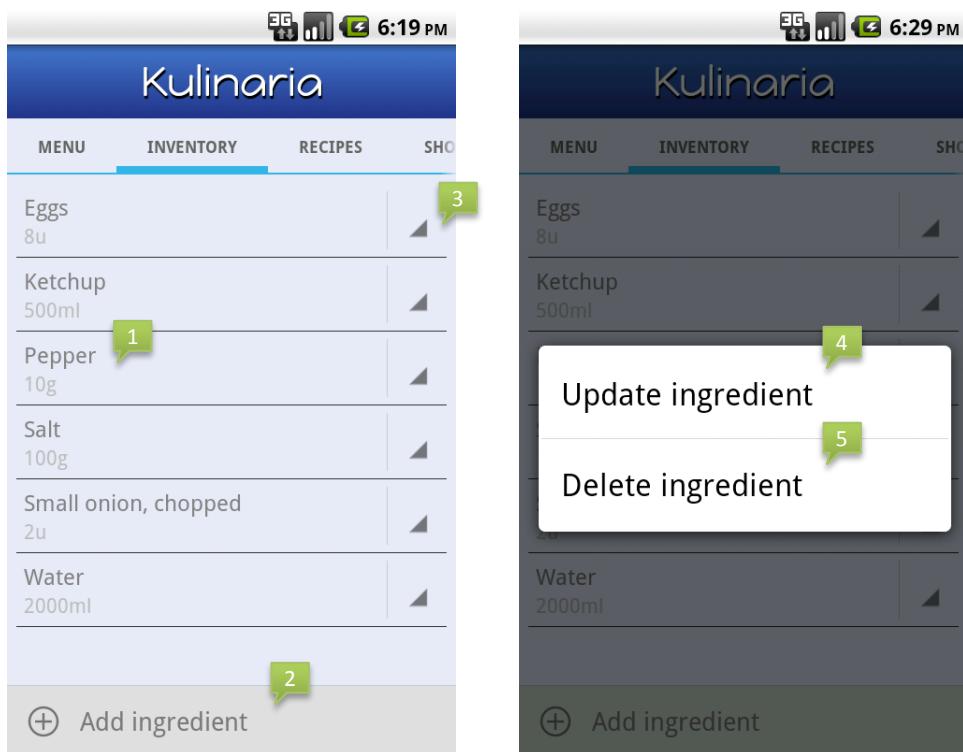


In seguito saranno esposte le caratteristiche di ogni pagina appartenente allo slider nello stesso ordine in cui sono state implementate, diverso dall'ordine in cui vengono presentate nell'applicazione. L'ordine delle pagine in Kulinaria è stato pensato al fine di rendere immediatamente disponibili le funzionalità principali, quelle che ipoteticamente saranno usate più di frequente, per aumentare l'usabilità.

INVENTORYPAGE

ASPETTO

In questa pagina sono memorizzati tutti i prodotti che l'utente possiede realmente nella propria cucina. È compito dell'utente tenere sempre aggiornata la lista degli ingredienti per ottenere risultati corretti e coerenti con le reali disponibilità alimentari.



1. LISTA DEGLI INGREDIENTI

Una ListView all'interno del layout della pagina mostra l'elenco dei prodotti nell'inventario. I prodotti sono memorizzati in un database locale all'applicazione e gestito dalla classe `DatabaseInterface`. Ogni prodotto ha un proprio nome, una quantità e un'unità di misura a cui si riferisce la quantità. La lista viene richiesta al gestore del database e ottenuta attraverso una matrice associativa di valori, più esattamente un `ArrayList<Map<String, Object>>`. Il riempimento della vista (qualora ci siano ingredienti memorizzati) avviene attraverso un `SimpleAdapter`⁴ che usa il layout definito nel file `inventory_list_item` per riempire ciascuna riga. All'adattatore viene poi associato un `SimpleAdapter.ViewBinder`, necessario per permettere l'associazione di un listener all'evento `onClick` del pulsante di opzioni .

⁴ Non è stato usato un `SimpleCursorAdapter` per l'assenza nella tabella `INVENTORY` del campo `_id` come chiave primaria intera e autoincrementante. Inizialmente era stato previsto questo tipo di adattatore, però dato che l'univocità degli ingredienti viene ottenuta dal solo nome, l'inserimento di un nuovo identificatore numerico non ha fatto altro che aumentare la ridondanza delle informazioni e dei riferimenti.

2. AGGIUNTA DI UN INGREDIENTE

L'aggiunta di un nuovo ingrediente avviene attraverso un form da compilare in una finestra di dialogo sovrastante l'activity. La creazione, la gestione del dialog, dei risultati e delle conseguenze derivanti dall'inserimento di un nuovo ingrediente vengono rimandate al metodo `showIngredientDialog()`.

3. OPZIONI ASSOCIATE AD OGNI INGREDIENTE

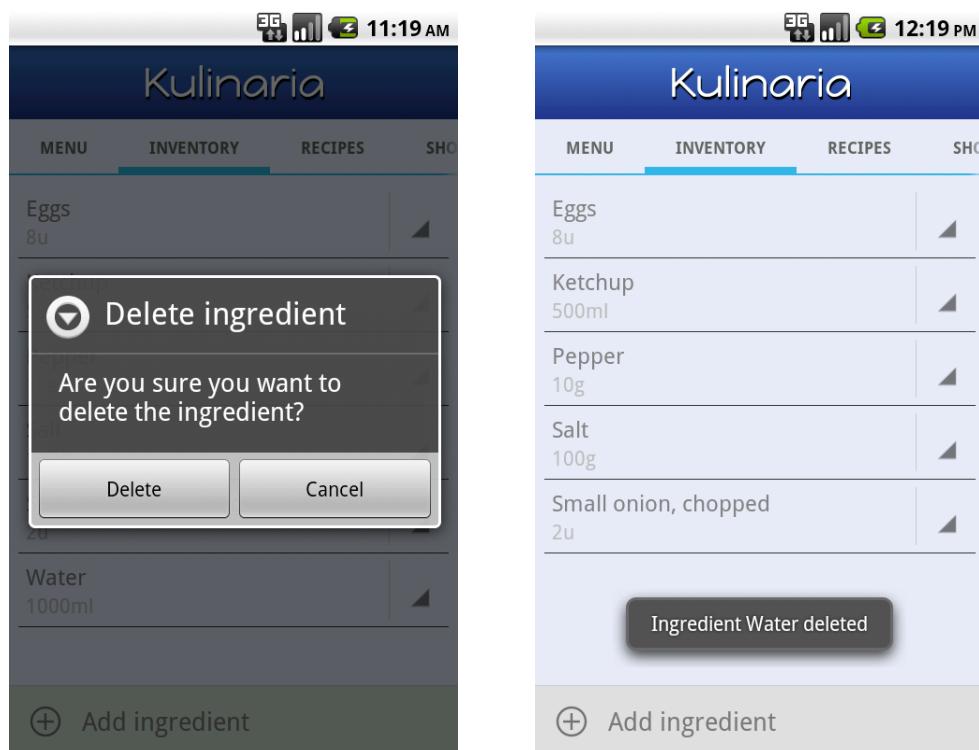
Le opzioni associate ad ogni ingrediente sono la semplice modifica e l'eliminazione dall'inventario. La scelta avviene attraverso una finestra di dialogo a scelta multipla (AlertDialog) che si apre quando viene premuto il pulsante di opzioni in ogni riga dell'inventario.

4. MODIFICA DI UN INGREDIENTE

Il click sul primo elemento della finestra di dialogo comporta la sola chiamata al metodo privato `showIngredientDialog()` della classe `MainActivity`. Grazie ad esso viene aperta una nuova finestra personalizzata in cui vengono mostrate tutte le informazioni riguardanti l'ingrediente in questione. È lo stesso metodo che viene chiamato quando si cerca di inserire un nuovo prodotto, ma in questo caso il parametro passato è il nome dell'ingrediente, in modo da distinguere l'inserimento dall'aggiornamento.

5. CANCELLAZIONE DI UN INGREDIENTE

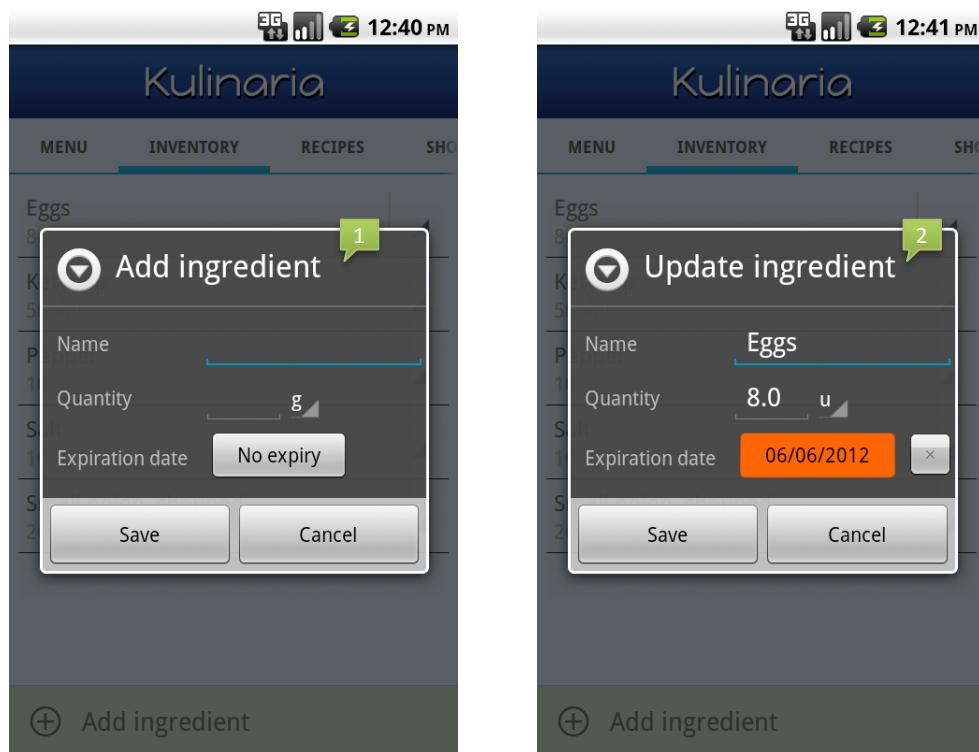
Nel rispetto dei criteri di usabilità generale, nell'istante in cui viene selezionata l'opzione per la rimozione di un ingrediente appare al contempo un alert per confermare la scelta effettuata (l'utente può aver premuto lo schermo senza volere). Qualora la scelta fosse ulteriormente confermata, il prodotto viene definitivamente rimosso dal database e l'utente viene notificato attraverso un pop-up di `Toast`.



SHOWINGREDIENTDIALOG

showIngredientDialog(String) è un metodo privato della classe MainActivity che si occupa di mostrare una finestra di dialogo per l'inserimento di un nuovo ingrediente e l'aggiornamento di un ingrediente selezionato già presente nell'inventario. In entrambi i casi, si sfrutta un layout comune definito nel file ingredient_dialog.

Nelle prime revisioni di Kulinaria, questo metodo era inglobato nella InventoryPage, in quanto si pensava venisse utilizzato unicamente in questa sezione. Più avanti è stato esteso l'utilizzo anche alla ShoppingListPage, causando un'esternalizzazione del metodo che lo rendesse visibile a tutte le pagine.



1. NUOVO INGREDIENTE

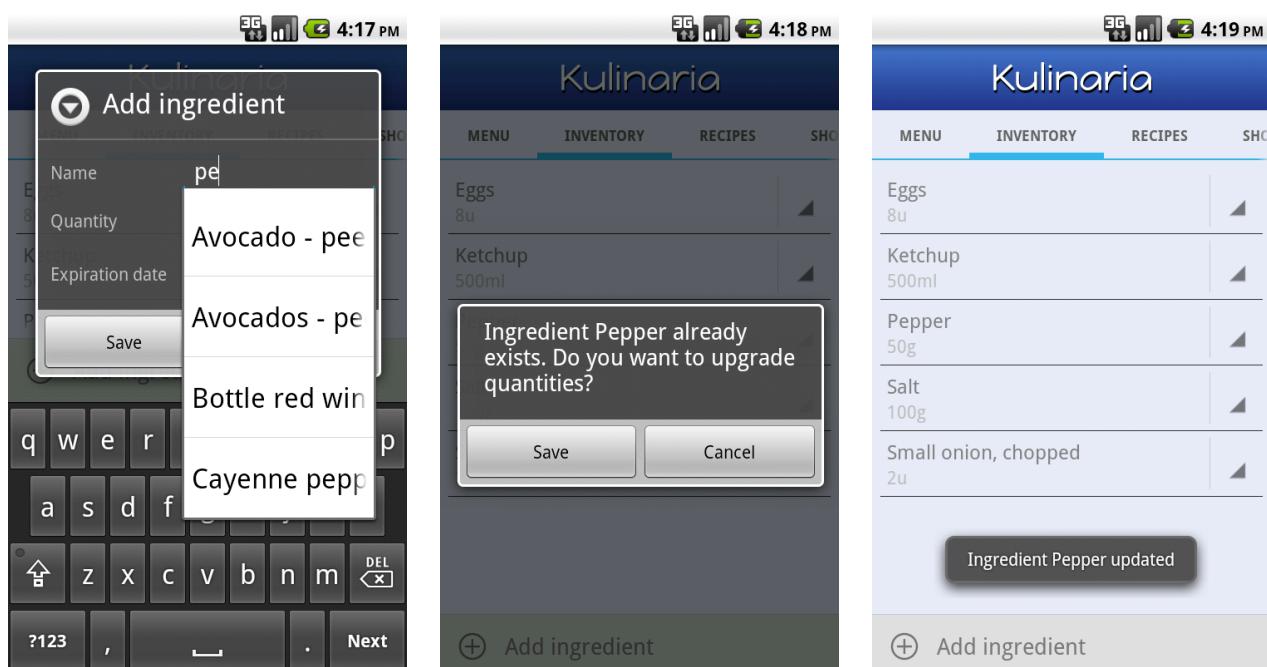
In questa schermata si permette l'inserimento di tutte le informazioni relative ad un nuovo ingrediente, ossia nome, quantità, unità di misura associata e data di scadenza. Essendo una delle parti più tediose da completare da parte dell'utente, si è cercato di aiutare quantomeno la digitazione del nome dell'ingrediente attraverso dei suggerimenti basati sull'elenco dei prodotti presenti nell'intero ricettario (vedi [RecipesPage](#)). Di conseguenza, il campo di testo è diventato un AutoCompleteTextView che si occupa di mostrare un menu verticale sottostante al campo stesso e attivato con l'inserimento di almeno due caratteri (ad esclusione di spazi e segni di tabulazione).

Per quanto riguarda la data di scadenza, il tutto viene regolato attraverso un pulsante. Quando questo viene premuto si apre una finestra di DatePickerDialog che mostra giorno, mese e anno dell'ultimo giorno di validità che si vuole dare al prodotto al prodotto. Attraverso la variabile calendar locale all'OnClickListener del pulsante si gestisce la data nella finestra per l'aggiornamento del testo del Button; quando la data di scadenza è inferiore a quella odierna il pulsante viene colorato di rosso chiaro per segnalare immediatamente all'utente che il prodotto non è più buono.

Ecco come si presenta il listener:

```
// Expiration date button listener
expirationDate.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Calendar calendar = Calendar.getInstance(); ←
        SimpleDateFormat dateFormat =
            new SimpleDateFormat(getString(R.string.simpleDateFormat));
        try { calendar.setTime(dateFormat.parse(expirationDate.getText().toString())); } catch (ParseException e) { }
        new DatePickerDialog(MainActivity.this, new OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view,
                int year, int monthOfYear, int dayOfMonth) {
                calendar.set(year, monthOfYear, dayOfMonth); ←
                expirationDate.setText(
                    DateFormat.format(getString(R.string.simpleDateFormat), calendar));
                if (calendar.before(Calendar.getInstance()))
                    expirationDate.getBackground().setColorFilter(
                        new PorterDuffColorFilter(Color.rgb(255, 102, 0),
                            PorterDuff.Mode.SRC_ATOP));
                else expirationDate.getBackground().clearColorFilter();
                removeExpirationDate.setVisibility(View.VISIBLE);
            }
        }, calendar.get(Calendar.YEAR),
        calendar.get(Calendar.MONTH),
        calendar.get(Calendar.DAY_OF_MONTH)).show(); ←
    }
});
```

Una volta immessi tutti i campi nel modulo si procede con la memorizzazione: solo i prodotti con nome non nullo e quantità maggiore di zero vengono salvati in database. Inoltre, per garantire l'univocità del nome all'interno dell'intera lista dei prodotti occorre fare un controllo preventivo sul nome digitato; nell'eventualità che venga immesso un nome già presente, apparirà una nuova finestra di conferma che informa sull'esistenza del prodotto e chiede se si desidera sostituirlo aggiornando la quantità ad esso associata. Segue un'ultima finestrella Toast per la notifica dell'avvenuto inserimento/aggiornamento.



2. MODIFICA INGREDIENTE

L'aggiornamento di un ingrediente nell'inventario avviene sempre attraverso la chiamata al metodo `showIngredientDialog()`. La differenza in questo caso sta nel parametro che viene passato alla funzione, ovvero il nome del prodotto da modificare. Solo in questo caso vengono ricercate (attraverso l'oggetto `DatabaseInterface`) le informazioni collegate al prodotto, in modo da riempire il form e compilarlo con i valori corretti.

Nella modifica di un ingrediente è stato eliminato l'adattatore per l'autocompletamento del nome, perché ritenuto pressoché inutile. Il nome è già inserito nella `TextView` e lo scenario tipico di utilizzo di questo modulo si concentra solo sull'aggiornamento delle quantità di prodotto possedute, raramente si arriva a modificare il nome dell'ingrediente (al massimo per correggere errori di scrittura errata). Nell'ipotesi in cui questo accada, bisogna verificare che un prodotto con lo stesso nome non sia già presente nell'inventario: ciò comporterebbe una perdita del vecchio prodotto (con il vecchio nome) ed una sovrascrittura delle quantità del prodotto già presente, solo dopo aver confermato la scelta nella finestra di dialogo simile a quella appena vista nell'aggiunta di un ingrediente.

REFRESH

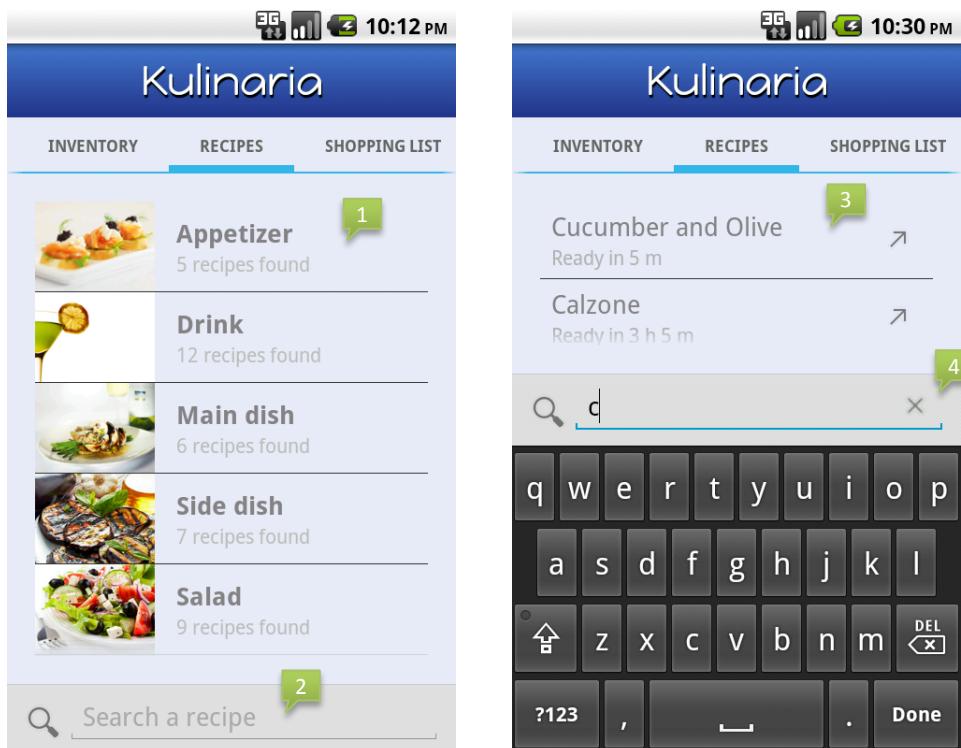
Ogni volta che viene inserito un nuovo prodotto, modificato o cancellato un prodotto esistente è necessario aggiornare la lista nell'inventario. Questo compito è apparentemente affidato al metodo pubblico `refresh()`, ereditato e richiesto dalla classe `Page`.

```
@Override  
public void refresh() {  
    layout.removeViewAt(0);  
    layout.addView(getInventoryListViewFromDatabase(), 0);  
}
```

Come si nota, il metodo non fa altro che eliminare il primo elemento della pagina e inserire una nuova vista al suo posto. La vista in questione viene ottenuta da `getInventoryListViewFromDatabase()` che restituisce una `ListView` solo se sono stati memorizzati prodotti nell'inventario, viceversa restituisce una `TextView` che ne notifica l'assenza.

RECIPESPAGE

La RecipesPage è la pagina dello slider dedicata al ricettario. In questa sezione è possibile visionare tutte le ricette memorizzate in database, sfogliando le varie categorie o ricercandole direttamente.



1. LISTA DELLE PORTATE

Appena si fa scorrere l'activity verso questo Fragment, viene mostrato l'elenco delle portate, ognuna delle quali ha associato un'immagine e un nome, memorizzati staticamente all'interno dell'applicazione, rispettivamente nella cartella drawable e nel file arrays. I piatti attualmente esistenti sono *Appetizer* (antipasto), *Drink* (bevute), *Main dish* (piatto principale), *Side dish* (contorno), *Salad* (insalatona) e *Dessert* (dolci).

La categoria sotto la quale viene raggruppata una ricetta è data per l'appunto dal piatto di portata a cui viene associata e per ogni categoria nell'elenco viene mostrato anche il numero di ricette totali associate a quel piatto e memorizzate in database.

Affinché fosse possibile riempire una ListView con oggetti statici e di natura mista (testo e immagini) è stato creato un vettore di array associativi opportunamente riempiti, al fine di sfruttarlo per definire un semplice SimpleAdapter su uno schema di riga definito dal file dishes_list_item. Le immagini relative ad ogni portata sono contenute nella cartella drawable-XX-hdpi, dove XX rappresenta il codice internazionale della lingua locale impostata nel dispositivo:

```
// Filling contents using SimpleAdapter
ArrayList<Map<String, Object>> dishes = new ArrayList<Map<String, Object>>();
```

```

String[] dishesList = getResources().getStringArray(R.array.dishes);
for (String dish : dishesList) {
    Map<String, Object> dishInfo = new HashMap<String, Object>();
    dishInfo.put("image", getResources().getIdentifier(
        dish.toLowerCase().replace(" ", "_"), "drawable", getPackageName()));
    dishInfo.put("name", dish);
    dishInfo.put("other", getString(R.string.recipesNumber)
        .replaceFirst("\\?", Integer.toString(db.getRecipeCount(dish))));
    dishes.add(dishInfo);
}
list.setAdapter(new SimpleAdapter(MainActivity.this, dishes, R.layout.dishes_list_item,
    new String[] { "image", "name", "other" },
    new int[] { R.id.listDishImage, R.id.listDishName, R.id.listDishOther }));

```

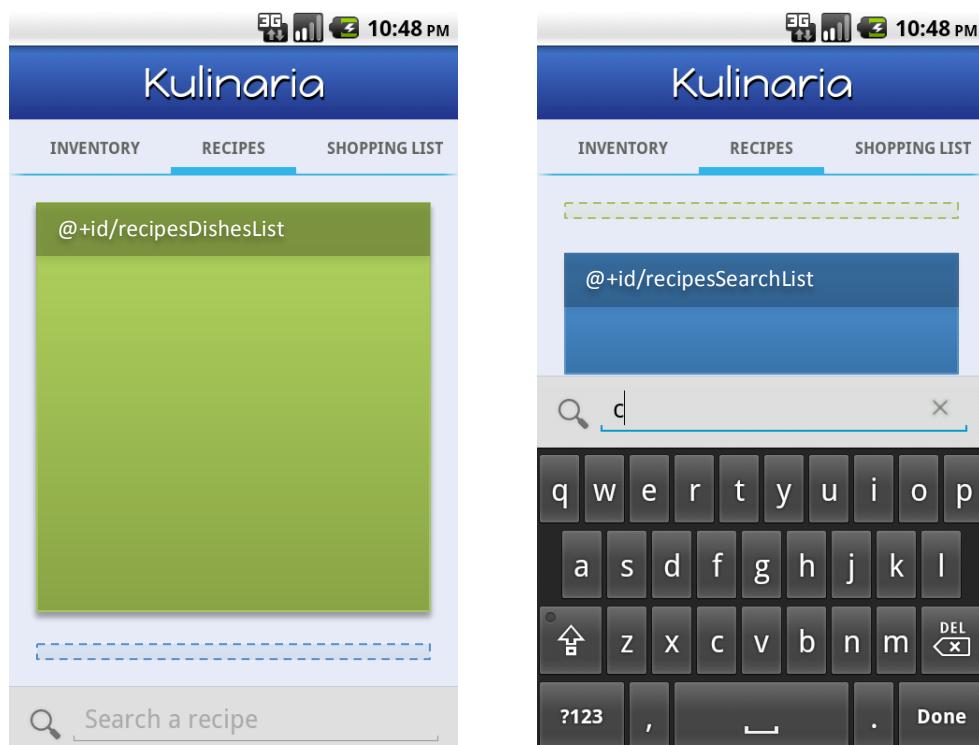
Ogni elemento della lista è collegato ad un Intent (grazie ad un ViewBinder) che permette di lanciare l'activity `RecipesListActivity` che visualizza tutte le ricette appartenenti a quella categoria scelta.

2. RICERCA RICETTA

Un altro aspetto all'interno del layout che salta immediatamente all'occhio è il campo di ricerca posizionato in basso. Esso rappresenta un ottimo metodo per trovare in poco tempo una ricetta fra tutte quelle inserite, indipendentemente dalla categoria d'appartenenza. I risultati vengono visualizzati al posto delle portate durante l'inserimento del nome nella TextView.

3. RISULTATI DELLA RICERCA

Fin dal primo inflate, la pagina è predisposta a contenere una particolare ListView per i risultati di ricerca. Inizialmente si parte da uno stato in cui la lista è nascosta e priva di elementi: una volta che il listener associato al campo di ricerca si attiva, la vista delle portate viene nascosta e viene visualizzata quella dei risultati, così da dare l'impressione all'utente finale che sia un'unica lista a modificarsi aggiornandosi alle sue esigenze.



In questo caso, ogni elemento della lista di ricerca permette di lanciare un'altra activity, chiamata [Recipe-Activity](#) nella quale è possibile visualizzare tutte le informazioni riguardanti la ricetta scelta.

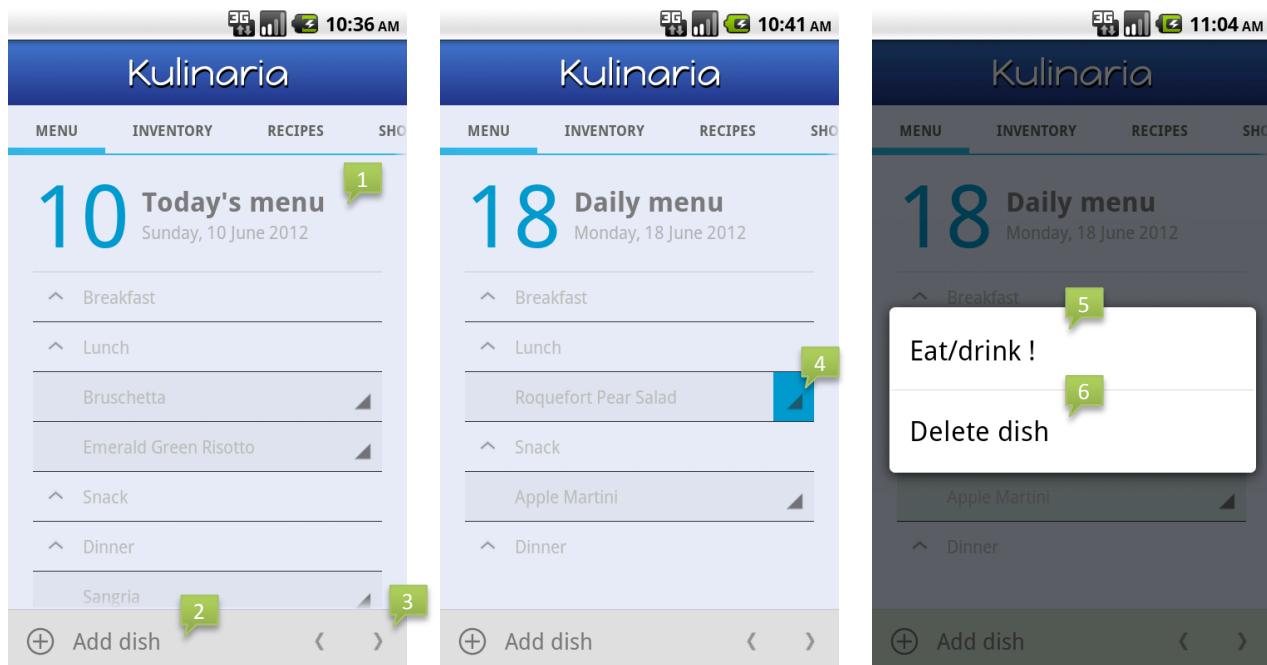
4. RESET DEI RISULTATI DELLA RICERCA

Durante la ricerca di una ricetta è possibile notare la presenza di una “X” nella parte laterale destra del campo di testo. Così come suggerisce l'intuito, quando l'immagine viene premuta il campo si riazzera e si riporta l'intera pagina alla situazione iniziale: la lista di ricerca torna ad essere nascosta, al contrario di quella delle portate che occupa l'intero spazio rimanete all'interno del layout.

L'immagine per il text-clearing non è esterna al campo di ricerca, bensì viene dinamicamente impostata (e rimossa) come parte integrante della TextView nella parte destra con il metodo `setCompoundDrawables()`.

MENUPAGE

La pagina dei menù è la prima schermata che appare con l'apertura della MainActivity e si può considerare una tra la funzionalità più utilizzate dall'utente. La struttura è semplice e mostra giorno per giorno la programmazione alimentare organizzata per pasto.

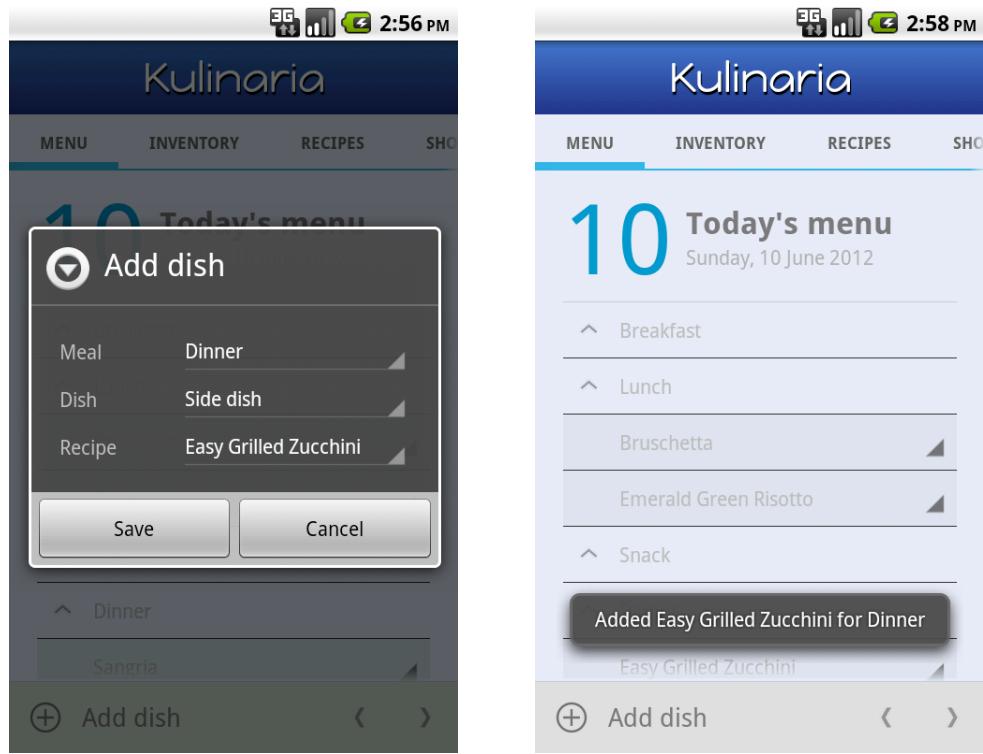


1. HEADER PRINCIPALE

Nella testata è visualizzato in ogni istante il giorno di riferimento del quale si sta osservando il menù. Essa comprende data in formato numerico, il giorno della settimana, il mese e l'hanno, oltre ad un'informazione più specifica che indica se ci stiamo riferendo al menù odierno o meno.

2. AGGIUNTA DI UNA PIETANZA

Nella parte laterale sinistra della bottom bar è stato posizionato il pulsante per l'aggiunta di una ricetta al menù. Quando viene premuto si apre una finestra di dialogo che permette di scegliere quale piatto si desidera pianificare per un determinato pasto (colazione, pranzo, ...). Le ricette, prese dal ricettario memorizzato nella base di dati (vedi [RecipesPage](#)), sono organizzate e filtrate per portata (antipasto, contorno, dolce, ...) e sono univoche per un determinato pasto del giorno. Ciò vuol dire ad esempio che non è possibile avere lo stesso contorno due volte per pranzo, oppure lo stesso dessert più di una volta per cena. Infatti, le scelte fatte e presenti nel menù non appaiono più nella scelta multipla della finestra dialogo, negando di fatto la possibilità di violare tale vincolo.



3. CAMBIAMENTO DEL GIORNO DI RIFERIMENTO

Sempre nella bottom bar, nella parte destra, sono presenti altri due ImageButton atti a permettere di cambiare il giorno visualizzato. Grazie al metodo privato MenuPage.inflateView() e la variabile day (di tipo Date) anch'essa privata alla classe della pagina, il listener per questi oggetti risulta estremamente semplice da realizzare:

```
// Change date buttons5
((ImageButton) layout.findViewById(R.id.menuDayBefore)).setOnClickListener(
    new OnClickListener() {
        @Override public void onClick(View view) {
            day.setDate(day.getDate()-1);
            inflateView(layout);
        }
    });
((ImageButton) layout.findViewById(R.id.menuDayAfter)).setOnClickListener(
    new OnClickListener() {
        @Override public void onClick(View view) {
            day.setDate(day.getDate()+1);
            inflateView(layout);
        }
    });
}
```

In inflateView() si fa riferimento a day, perciò una volta aggiornato il suo valore si modificano tutti i campi nell'header e si reimposta l'adattatore della lista attraverso il metodo inflateMenuMealsList().

⁵ layout rappresenta il riferimento all'oggetto LinearLayout che identifica il blocco principale della pagina dei menù.

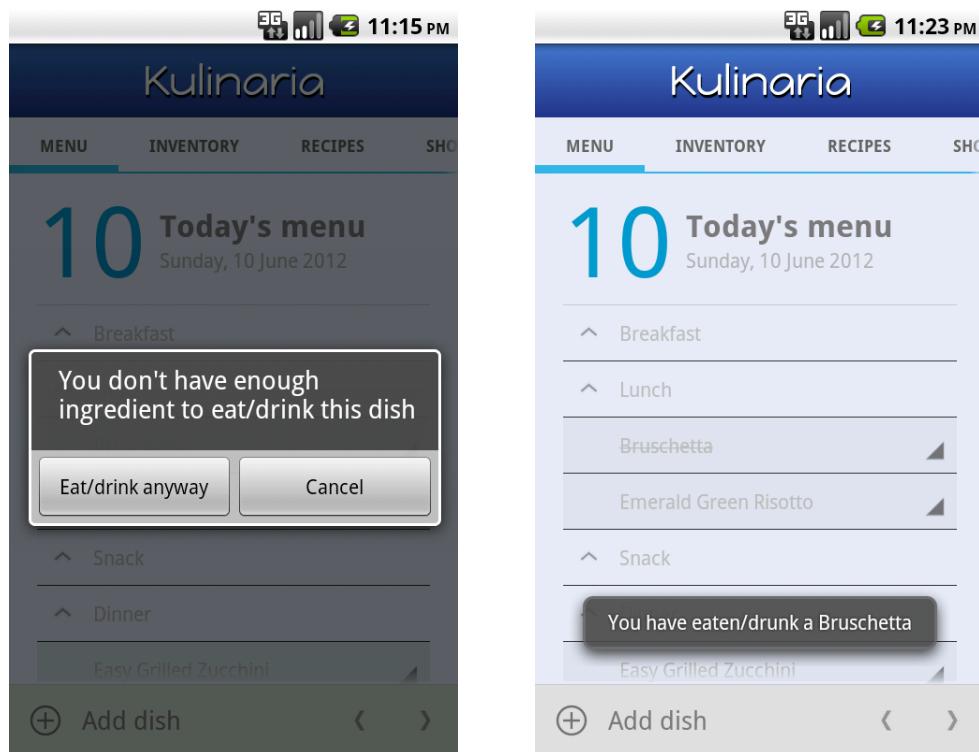
4. OPZIONI ASSOCIATE AD OGNI PASTO

Ogni elemento della lista nella pagina, ad esclusione delle etichette di raggruppamento, presenta un pulsante simile a quello visto in [InventoryPage](#). Quando viene cliccato si apre un'AlertDialog a scelta multipla che permette di notificare la consumazione o scegliere l'eliminazione del piatto.

5. CONSUMAZIONE DI UN PASTO

Grazie al primo elemento della finestra è possibile impostare una pietanza come "mangiata/bevuta". Così facendo vengono rimossi dall'inventario tutti i prodotti necessari alla preparazione, la ricetta viene barrata con una linea orizzontale sopra il testo e non viene più considerata nel calcolo degli ingredienti che generano la shopping list (vedi [ShoppingListPage](#)). Più precisamente, ciò che accade è il set di un flag apposito nella tabella dedicata ai menù, con conseguente creazione di una nuova ListView che sostituisce quella esistente attraverso il semplice richiamo al metodo `inflateMenuMealsList()`.

Naturalmente per terminare un pasto è necessaria la presenza nell'inventario di tutti i prodotti necessari per preparare le ricette selezionate. Nonostante ciò, può capitare che l'utente non mangi solo e che siano altre persone a portare gli ingredienti mancanti per suo conto, oppure ha deciso di preparare la ricetta con un numero inferiore di ingredienti, ad esempio per seguire una dieta che gli è stata prescritta (senz'olio, senza sale, ...). Per questo motivo si deve permettere la notifica del compimento del pasto e l'eliminazione del sottoinsieme di prodotti dell'inventario che saranno ugualmente impiegati, attraverso una finestra di dialogo che informa della mancanza degli ingredienti e permette di scegliere se consumarli.



6. CANCELLAZIONE DI UN PASTO

Il secondo elemento della finestra di dialogo serve a eliminare un pasto dall'elenco, indipendentemente che esso sia stato già consumato o meno. Prima dell'eliminazione si presenta la solita schermata che chiede conferma per l'azione irreversibile che si vuole intraprendere.

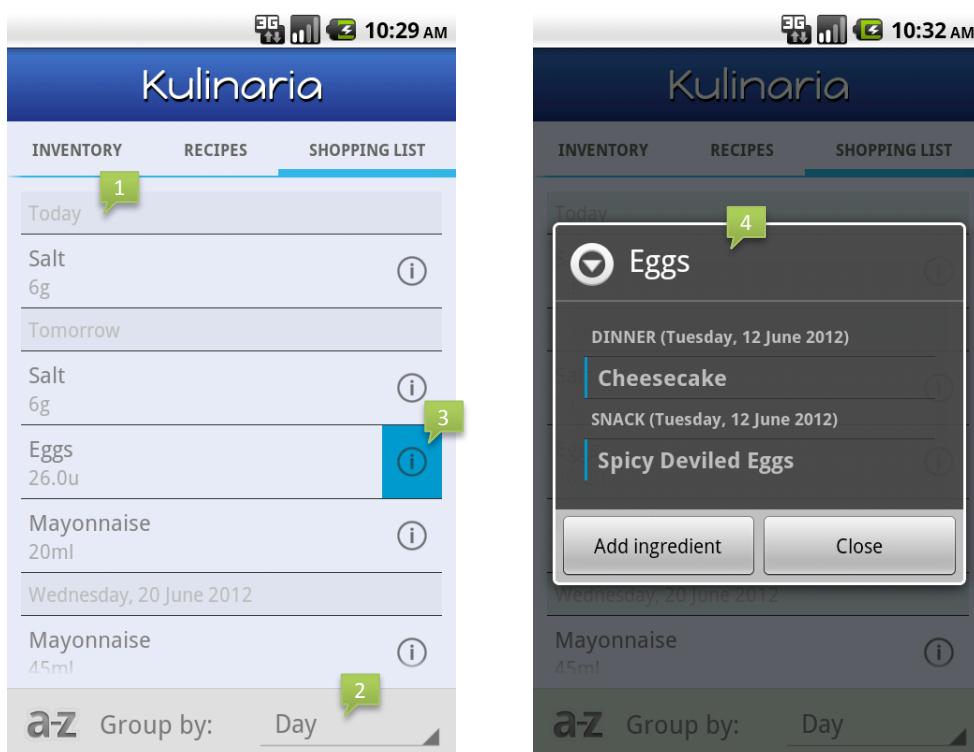
SHOPPINGLISTPAGE

ASPETTO

Un altro componente essenziale di Kulinaria è la shopping list, la lista degli ingredienti che mancano all'utente per poter preparare le ricette programmate per i giorni successivi a partire da quello corrente.

Originariamente era stata prevista una tabella dedicata in database, atta a supportare questa funzionalità. In seguito si è capito che ogni modifica all'inventario o alla lista dei menù comporta un aggiornamento della shopping list, perciò un aggiornamento sia della tabella che dell'elenco nella vista. In questo modo si rendevano necessari due cicli per garantire il funzionamento della pagina, uno per il calcolo degli ingredienti e contemporaneo *aggiornamento* del database (oppure, invece della modifica, attraverso l'invalidazione dei vecchi e l'inserimento di quelli nuovi) ed uno per il recupero dei dati e la ridefinizione dell'adattatore per la ListView.

Pertanto si è deciso di unire le due operazioni al fine di alleggerire il carico computazionale necessario, semplicemente eliminando lo schema in database. Tutte le volte che si verifica un cambiamento agli elementi che causano la modifica della shopping list, viene inoltrato l'aggiornamento della pagina attraverso il metodo refresh() (ereditato da Page). In questo caso non si fa altro che sostituire la pagina con una nuova istanza rigenerata, in cui la lista viene ricalcolata in funzione delle nuovi variabili in gioco. Non esiste dunque alcun riferimento statico e permanente alla shopping list nell'applicazione, bensì si opera il ricalcolo della lista dei prodotti mancanti ognqualvolta ne viene richiesta l'esigenza.



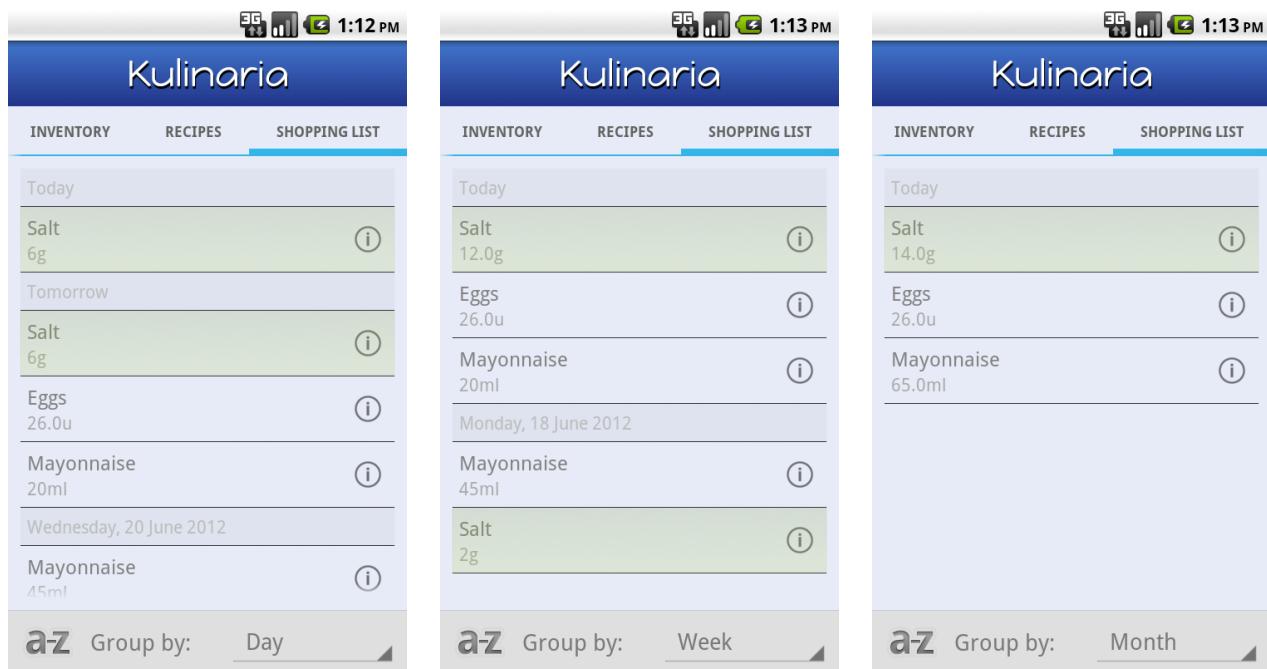
1. LISTA DEI PRODOTTI

Incrociando l'elenco degli ingredienti necessari a preparare le ricette scelte nel menù e l'elenco degli ingredienti che sono già presenti nell'inventario, si ottiene la lista dei prodotti mancanti. Per ottenere una lista coerente è necessario controllare anche le quantità mancanti e le relative unità di misura. Qualora si verificasse un'incongruenza tra l'unità di misura indicata nella ricetta e quella scelta nell'inventario allora il prodotto è da considerarsi completamente mancante. L'aggiornamento e la verifica della correttezza delle informazioni nell'inventario sono affidate all'utente.

In questo caso, la lista viene modellata attraverso un `GroupedListAdapter` che permette di inserire facilmente delle intestazioni con cui suddividere gli elementi (vedi [GroupedListAdapter](#)).

2. ORGANIZZAZIONE DEI PRODOTTI

Ovviamente la maggior parte delle persone non fa la spesa ogni giorno. Per questo motivo Kulinaria prevede tre viste differenti per la shopping list: per giorno, per settimana e per mese. Ogni vista si riferisce al raggruppamento con cui sono rappresentati i vari prodotti; ad esempio se il sale serve oggi per 6 g (grammi) e domani per altri 6 g, nella vista settimanale appariranno 12 g di sale totale per portare a termine le ricette scelte.



Si faccia attenzione che il giorno in etichetta nella vista settimanale e mensile si riferisce al primo giorno a cui tale vista fa riferimento. Nella figura precedente, Lunedì 18 giugno è il primo giorno della settimana successiva a quella attuale, però la ricetta a cui si fa riferimento è programmata per il giorno 20 (si veda la vista giornaliera). Allo stesso modo, nella vista mensile l'unico gruppo presente è da intendere "da oggi" fino ai prossimi 30 giorni.

3. INFORMAZIONI ASSOCIATE AD OGNI PRODOTTO

Ogni elemento nella shopping list segue il layout descritto nel file `shopping_list_item`: in esso sono posizionate tre `TextView` per identificare il nome, la quantità e l'unità di misura dell'ingrediente, più

un'inequivocabile immagine che rappresenta la lettera "i", per permette di visualizzare tutte le informazioni relative all'ingrediente considerato.

4. RIEPILOGO DELLE INFORMAZIONI

A primo impatto, la shopping list mostra solo gli ingredienti che mancano per completare i piatti scelti nella pagina dei menù. Maggiori dettagli si possono avere cliccando sull'immagine laterale, grazie alla quale vengono visualizzate tutte le informazioni necessarie a comprendere il motivo per cui un ingrediente si trova nella lista. Più precisamente, vengono mostrate tutte le ricette, con relativa data scelta e pasto programmato, che hanno bisogno di quel prodotto per essere realizzate. L'adattatore che viene utilizzato in questo caso è dello stesso tipo della shopping list, `GroupedListAdapter`, ciò che cambia sono i contenuti ed il layout per ogni elemento figlio di una sezione (`shopping_list_dialog_item`). Dalla stessa finestra di dialogo è possibile inoltre aggiornare l'inventario e aggiungere direttamente l'ingrediente selezionato: il tutto avviene attraverso la chiamata al metodo `showIngredientDialog()`, che verifica la presenza o l'assenza del prodotto e apre, a seconda del caso, una finestra per l'inserimento o per l'aggiornamento. Una volta terminato il suo compito, vengono aggiornate sia la pagina dei menù che quella della shopping list, attraverso le seguenti chiamate:

```
// Updating the inventory page  
inventoryPage.refresh();  
// Updating the shopping list page  
shoppingListPage.refresh();
```

Le due variabili che puntano alle pagine sono locali e private all'activity, così come già accennato in [MainActivity](#).

GROUPEDLISTADAPTER

L'aspetto di una `ListView` che usa un adattatore di questo tipo è molto simile all'`ExpandableListView` presente in [MenuPage](#), anche se sostanzialmente molto differente. La principale differenza sta nel fatto che il primo è composto da header che collassano e espandono la sezione che li segue quando vengono cliccati, mentre il secondo presenta degli header fissi, statici, per puro scopo informativo che non permettono di nascondere le liste che le susseguono.

Ogni sezione è identificata da una stringa di testo che determina anche l'intestazione e da un adattatore semplice con cui riempire la sezione. L'idea consiste nell'usare una lista con un numero di elementi pari alla somma degli elementi in ciascuna sezione e degli header relativi (ad esempio, se ci fossero 4 sezioni con 5 elementi ciascuna, la lista sarebbe preallocata esattamente con $4 + (5 \times 4) = 24$ elementi totali). In seguito si restituisce una vista personalizzata sulla base dell'entità da restituire, sia essa un titolo o un elemento di una sezione.

Anche l'uso è molto semplice: ogni sezione viene aggiunta col metodo `addSection()` e permette di specificare la stringa per l'intestazione e l'adattatore associato all'elenco che viene poi memorizzato nell'array associativo privato dell'oggetto.

Quanto segue illustra l'uso dell'adattatore nella finestra di dialogo che appare cliccando sull'immagine delle informazioni relative ad un ingrediente nella shopping list.

```

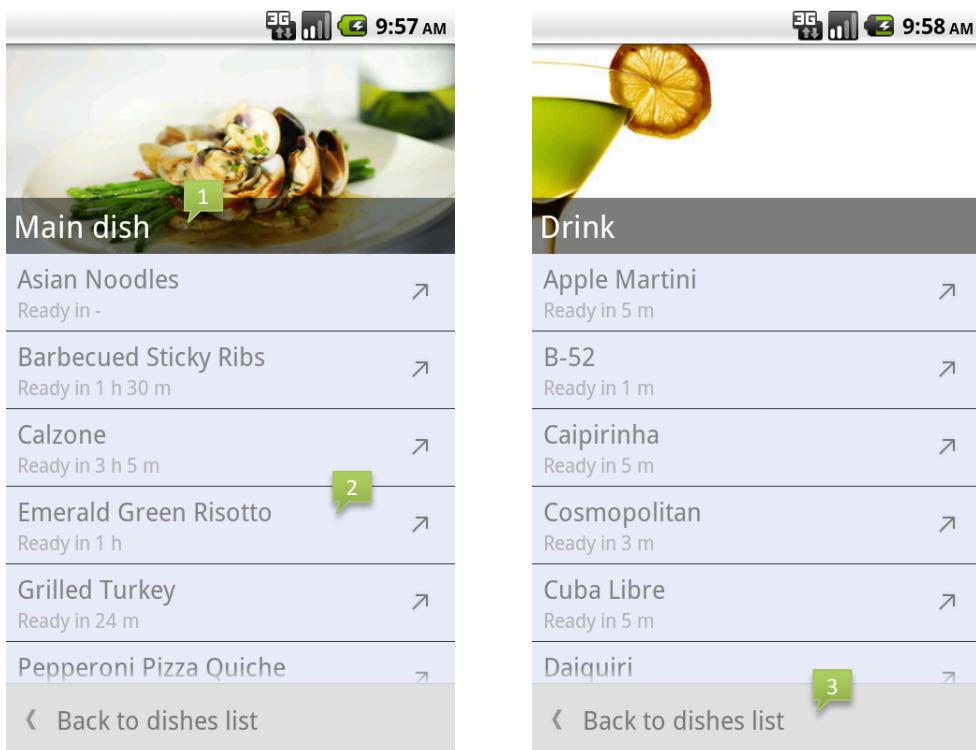
// Fill the ListView
GroupedListAdapter shoppingListAdapter = new GroupedListAdapter(
    MainActivity.this, R.layout.shopping_list_dialog_header);
for (int m = 0; m < meals.length; m++) {
    Date date = null;
    SimpleDateFormat format = new SimpleDateFormat(DatabaseInterface.DATAFORMAT);
    try { date = format.parse(dates[m]); }
    catch (ParseException e) { }
    shoppingListAdapter.addSection(meals[m].toUpperCase() + " (" +
        new SimpleDateFormat(getString(R.string.extendedDateFormat)).format(date)+ ")",
        new ArrayAdapter<String>(MainActivity.this,
            R.layout.shopping_list_dialog_item, new String[] { recipes[m] }));
}
((ListView)
dialogView.findViewById(R.id.shoppingListDialogList)).setAdapter(shoppingListAdapter);

```

Nell'esempio meals, dates e recipes rappresentano rispettivamente i vettori contenenti il nome dei pasti della giornata, il giorno di riferimento e la ricetta di cui è richiesta una certa quantità dell'ingrediente.

RECIPESLISTACTIVITY

Dalla pagina `MainActivity.RecipesPage` si possono visionare tutti i tipi di pietanza presenti all'interno di Kulinaria. Premendo su una delle portate in elenco si attiva un Intent che permette il lancio dell'activity `RecipesListActivity`, nella quale vengono mostrate tutte le ricette per una specifica categoria⁶.



1. INTESTAZIONE

Nella parte superiore della schermata si presenta un'immagine illustrativa della tipologia di ricetta insieme al nome. Questa sezione è stata realizzata attraverso un blocco `RelativeLayout` che risulta particolarmente vantaggioso soprattutto per le prossime implementazioni di Kulinaria. In seguito infatti, verrà data la possibilità all'utente di aggiungere, rimuovere e modificare le ricette memorizzate (attualmente queste funzionalità non sono supportate, vedi [Aggiunta/modifica/eliminazione ricette](#)) per mezzo di un pulsante per le opzioni nella parte laterale destra della striscia nera in trasparenza. Al momento compare solo il nome della portata attraverso una `TextView`, ma sarà modificata in un layout composito riempito da un campo di testo e da un'immagine.

⁶ Il nome della portata viene passato come extra dell'intent:

```
startActivity(new Intent(MainActivity.this, RecipesListActivity.class)
    .putExtra("dish", ((TextView) view.findViewById(R.id.listDishName)).getText()));
```



2. ELENCO DELLE RICETTE

La lista delle ricette è costruita con lo stesso procedimento utilizzato per la ricerca di un ingrediente nella RecipesPage dello slider (vedi [Risultati della ricerca](#)). Anche in questo caso all'utente viene fornito un feedback immediato sul tempo di preparazione (totale) necessario a completare la ricetta, corredato da una freccia per il richiamo all'azione di visualizzazione. In realtà il listener è collegato all'evento di click sull'intera riga e permette di avviare l'activity [RecipeActivity](#) per vedere le informazioni specifiche relative all'elemento cliccato.

3. BACK BUTTON

Nella parte inferiore dell'activity è stato sistemato un pulsante per tornare alla schermata precedente, ossia la MainActivity, l'unica che può portare l'utente a questa interfaccia. Non viene però creato un nuovo intent che avvia l'activity precedente, che sarebbe vista come un nuova schermata nella storia di navigazione dell'utente, bensì viene chiamato il metodo `finish()` che chiude l'attività corrente.

```
((Button) findViewById(R.id.dishBackButton)).setOnClickListener(new View.OnClickListener(){
    @Override public void onClick(View view) { finish(); }
});
```

RECIPEACTIVITY

Questa schermata molto semplice mostra tutte le informazioni relative ad una ricetta memorizzata nel database dell'applicazione.



1. TOP BAR

La parte alta dell'activity è caratterizzata dal nome della ricetta che si sta visionando e da un pulsante per tornare all'activity precedente. In questo caso, l'activity precedente può non essere RecipesListActivity, poiché un altro riferimento a questa "pagina" può essere lanciato attraverso un intent nella scheda RECIPES della MainActivity (vedi [Risultati della ricerca](#)).

Per quanto riguarda il titolo, invece, è stato utilizzato un oggetto di tipo AutoResizeTextView che estende TextView e permette di ridimensionare il font del campo di testo quando il contenuto supera la dimensione che gli è stata assegnata⁷.

2. DESCRIZIONE DELLA RICETTA

Le informazioni relative alla ricetta sono tutte prese dal database e si riferiscono a nome del pasto, tempo di preparazione, tempo totale, numero di persone, ingredienti e descrizione dettagliata della preparazione.

⁷ Maggiori informazioni su <http://stackoverflow.com/questions/5033012/auto-scale-textview-text-to-fit-within-bounds>.

DATABASE INTERFACE

La classe DatabaseInterface è sicuramente fra le più importanti per il corretto funzionamento dell'applicazione, è quella che si occupa di gestire tutte le informazioni e i dati associati al database fornendo un'interfaccia semplice e trasparente rispetto all'implementazione. Tutte le query vengono fatte all'interno dei suoi metodi, ciò che viene esternalizzato è la rappresentazione del dato richiesto con un aspetto più consono alla richiesta fatta (ad esempio attraverso valori interi, booleani o stringa), oppure tramite un riferimento all'insieme dei risultati ottenuti con un oggetto di tipo Cursor. Laddove possibile, si è sempre cercato di aprire e chiudere i cursori dentro i metodi di DatabaseInterface, cosicché ogni riferimento viene chiuso senza portare al crash dell'applicazione. Per questo motivo sono stati realizzati i metodi cursorToMapArray() e cursorToMapArrayIndexed() che identificano l'intero contenuto della tabella attraverso oggetti di tipo ArrayList<Map<String, Object>>.

IL DATABASE

Android sfrutta una versione semplificata di sql (SQLite3) che memorizza i dati all'interno di un file nella cartella privata dell'applicazione. La sicurezza è affidata al file system che garantisce la non accessibilità attraverso i permessi di lettura, scrittura e esecuzione (tipici dei s.o. unix-based) registrati sul file e sulla cartella.

GLI SCHEMI

Di seguito sono riportate le tabelle dell'applicazione sulla base di quanto detto fino a questo punto.

INVENTORY

| | | | | |
|------|------|----------|------|----------------|
| nome | name | quantity | unit | expirationDate |
| tipo | text | float | text | long |

In questa tabella sono memorizzati i prodotti dell'inventario che l'utente dovrebbe fisicamente possedere. I campi riguardano rispettivamente il nome del prodotto, la quantità, l'unità di misura e la data di scadenza. Il nome è anche l'attributo chiave della tabella che identifica una tupla.

RECIPES

| | | | | | | | |
|------|-----|------|------|-----------------|-----------|----------|-------------|
| nome | _id | name | dish | preparationTime | readyTime | servings | description |
| tipo | int | text | text | int | int | int | text |

Per ogni ricetta all'interno di Kulinaria (vedi [Descrizione della ricetta](#)) viene salvato il nome, la pietanza a cui appartiene (antipasto, primo piatto, dolce, ...), il tempo di preparazione, il tempo di preparazione totale, il numero di persone e la descrizione della preparazione. La chiave primaria è data dal campo intero _id autoincrementante.

RECIPESINGREDIENTS

| | | | | |
|------|----------|------------|----------------|------|
| nome | recipeId | ingredient | ingredientNeed | unit |
| tipo | int | text | float | text |

Dato che ogni ricetta ha un numero variabile di ingredienti che non può essere fisso, ma potenzialmente di dimensione infinita, è stata definita la tabella RecipesIngredients in cui si associa l'id di ogni ricetta al nome di un ingrediente, la quantità e l'unità di misura relativa. Ogni ingrediente è presente una volta sola in una ricetta, pertanto la chiave primaria è data dall'unione di recipeId e ingredient.

MENU

| | | | | |
|------|------|------|----------|------------|
| nome | date | meal | recipeId | eatenDrunk |
| tipo | date | text | int | tinyint |

Per quanto riguarda la programmazione dei menù di ogni giornata, si sfrutta la tabella Menu. I campi che interessano in questo caso sono rispettivamente il giorno a cui si fa riferimento, il pasto della giornata (colazione, pranzo, cena, ...), l'identificatore della ricetta che si è scelta per quel pasto di quella giornata e un marcitore booleano che indichi se il pasto è stato già mangiato o meno. Dal momento in cui SQLite non prevede colonne con valori di tipo booleano, si imposta il campo come un intero e lo si usa nel range [0-1]. Il formato con cui viene memorizzata e con cui deve essere prelevata la data del menù è memorizzata e impostata nella variabile statica DATEFORMAT al valore yyyy-MM-dd.

RIFERIMENTI AGLI SCHEMI

Essendo utilizzate praticamente ovunque all'interno di Kulinaria, le tabelle della base di dati devono avere un riferimento statico richiamabile da ogni classe. La parte che più interessa è quella relativa ai nomi delle colonne di ogni schema, cosicché sia possibile creare a proprio piacimento adattatori per ListView o semplicemente prelevare le singole informazioni relative, ad esempio, agli ingredienti o alle ricette.

In quest'ottica, DatabaseInterface presenta 4 classi statiche e pubbliche con la stessa struttura:

```
// Tables definition
public static final class TABLE_NAME {
    static final String TABLE          = "TableName";
    static final String column1        = "column1";
    static final String column2        = "column2";
    static final String column3        = "column3";
    static final String ORDER_BY       = column1 + " ASC";
    static final String CREATE =
        "create table if not exists " + TABLE + " ( " +
        column1 + " typeColumn1 primary key, " +
        column2 + " typeColumn2, " +
        column3 + " typeColumn3 )";
}
```

Oggetti di questo tipo possono permettere di richiamare i loro attributi senza che l'oggetto stesso sia istanziato, ma solo attraverso un suo riferimento statico, per l'appunto. Con riferimento all'esempio precedente, è possibile sapere il nome reale del primo campo fuori dalla classe DatabaseInterface semplicemente richiamando DatabaseInterface.TABLE_NAME.column1 all'interno del package.

DBHELPER

DbHelper è una classe di supporto per la creazione e la gestione delle versioni del database. Essendo utilizzata solo da DatabaseInterface, si presenta come classe annidata che estende SQLiteOpenHelper fornito dal framework.

La prima volta che viene avviata l'applicazione sul dispositivo mobile (o meglio, ogni volta che si avvia una nuova installazione) viene richiamato il metodo onCreate() ereditato dalla classe padre astratta ed è in questo punto in cui le tabelle vengono definite. Nonostante sia stato previsto l'attributo CREATE per ogni classe associata ad una tabella, l'operazione che viene fatta non si limita alla chiamata al metodo executeSQL(), bensì si occupa anche di popolare il database con un insieme di ricette preimpostate. Le ricette vengono lette da un file XML contenuto nella cartella res/raw e organizzato come segue:

```
<?xml version="1.0" encoding="utf-8"?>
<database>
    <table name="Recipes">
        <row>
            <name>Pepper Jelly</name>
            <dish>Appetizer</dish>
            <preparationTime>2</preparationTime>
            <readyTime>2</readyTime>
            <servings>6</servings>
            <description>Spread jelly over the block of cream cheese.</description>
        </row>
        <!-- other recipes -->
    </table>
    <table name="RecipesIngredients">
        <row>
            <recipeId>1</recipeId>
            <ingredient>Cream cheese, softened</ingredient>
            <quantity>224</quantity>
            <unit>g</unit>
        </row>
        <row>
            <recipeId>1</recipeId>
            <ingredient>Mild pepper jelly</ingredient>
            <quantity>150</quantity>
            <unit>g</unit>
        </row>
        <!-- other ingredients -->
    </table>
</database>
```

Una volta letto il file, viene salvato il contenuto in una struttura dati più flessibile (Map<String, ArrayList>) che viene iterata per permettere l'inserimento dei dati attraverso insert().

Quando invece la struttura del database viene modificata e quindi aggiornata la versione attraverso la variabile statica DatabaseInterface.VERSION, viene richiamata la routine onUpgrade() dell'oggetto DBHelper. Per non perdere i dati memorizzati dall'utente è necessario aggiornare le tabelle alla nuova versione solo dopo aver fatto un backup dei dati presenti. Le operazioni vengono svolte attraverso onUpgradeTable() che si occupa rispettivamente di:

1. Modificare la tabella presente rinominandola con il prefisso "temp_"
2. Creare la struttura della nuova versione della tabella
3. Recuperare l'elenco delle colonne presenti in entrambe le versioni

4. Inserire i record presenti nella vecchia tabella in quella nuova
5. Rimozione della tabella temporanea

STRUTTURA DELL'APPLICAZIONE

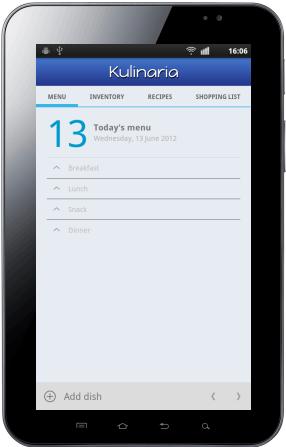
In base a quanto detto fino ad ora, si illustra la struttura di Kulinaria in termini di navigazione tra pagine dello slider e tra le activity. Si faccia attenzione al fatto che in ogni punto dell'applicazione è sempre presente (nonché necessario) il supporto al database offerto dall'oggetto DatabaseInterface.



ALTRO

DEVELOPING

Per lo sviluppo di questa prima versione di Kulinaria sono stati utilizzati diversi dispositivi, con prestazioni e risultati abbastanza differenti. In particolare, oltre all'AVD definito con i tool di supporto, lo sviluppo si è concentrato sui seguenti dispositivi:

| Dispositivo | HTC Wildfire S | Samsung Galaxy S II | Samsung Galaxy S II | Samsung Galaxy Tab |
|-------------------------|--|--|---|--|
| Versione | 2.2.1 | 2.3.3 | 4.0.3 | 2.3.3 |
| Risoluzione | 320x480 | 800x480 | 800x480 | 1024x600 |
| Screenshot applicazione |  |  |  |  |

REPOSITORY

Il repository dell'applicazione con le relative modifiche apportate ad ogni checkout è disponibile online all'indirizzo <https://github.com/indrimuska/Kulinaria>.

IMPLEMENTAZIONI FUTURE

Di seguito sono riportati i prossimi cambiamenti che subirà Kulinaria, alcuni necessari prima di essere registrata nel Google Play Store.

MULTILINGUA

Per come è stata realizzata l'applicazione, questa è sicuramente la modifica più semplice e più veloce da apportare. Si tratta infatti di creare un nuovo file strings.xml e arrays.xml in una cartella apposita per ogni lingua. Inoltre, siccome in ogni nazione sono presenti piatti particolari che la caratterizzano (ad esempio, in Italia un menù tipico è composto da primo, secondo e contorno), con la ridefinizione dell'array dishes, è

necessario inserire anche le nuove immagini per la pagina [RecipesPage](#) dello slider, che dovranno essere inserite nella cartella drawable-XX-xhdpi (XX è il codice della lingua) per la corretta visualizzazione.

AGGIUNTA/MODIFICA/ELIMINAZIONE RICETTE

Un'altra funzione di cui sarà dotata l'applicazione è l'aggiunta di nuove ricette. Chiunque deve poter aggiungere sia le ricette che usa più di frequente, che quelle un po' più particolari di cui è necessario vedere il procedimento. Anche per quelle che sono già presenti, l'utente deve essere messo in grado di modificarne sia il procedimento che le informazioni caratteristiche, nonché la possibilità di eliminarle totalmente.

Si può inoltre sfruttare la lettura del file XML per il riempimento del database al primo avvio dell'applicazione, mettendo a disposizione degli utenti pacchetti con liste di ricette già compilate. In questo modo altri possono distribuire attraverso la rete le proprie ricette e far sì che siano disponibili per tutti e importabili all'interno di Kulinaria (probabilmente il posto migliore è attraverso l'activity per le preferenze).

NOTIFICA DELLE SCADENZE

Un buon modo per sfruttare la data di scadenza inserita dall'utente è notificarne la vicinanza attraverso degli alert nella barra delle notifiche. Il miglior metodo è permettere attraverso le preferenze dell'applicazione la scelta del numero di giorni precedenti la data fissata entro i quali mostrare il messaggio e la frequenza di notifica di tali messaggi. Tramite un incrocio con il ricettario, è possibile anche suggerire all'utente un menù che sfrutti il maggior numero di ingredienti in scadenza in modo da permetterne la consumazione.

NOTIFICA PASTO

Quando arriva il momento del pranzo o della cena, una notifica nelle barre delle notifiche potrebbe avvisare l'utente e chiedergli se ha deciso di mangiare ciò che aveva scelto o se ha cambiato idea. Con la conferma della scelta, inizia tutta la routine di controllo già vista in [MenuPage](#), ovvero verifica della presenza di tutti gli ingredienti necessari, conferma dell'operazione da eseguire e aggiornamento delle due liste principali (inventario e shopping list).

INTEGRAZIONE CALENDARIO

Oltre alla solita vista dei menù dello slider, è possibile integrare lo storico delle scelte fatte e di quelle da fare insieme al content provider del calendario di sistema. Integrando gli eventi in un'agenda apposita e impostando automaticamente gli avvisi associati, si possono semplificare le implementazioni delle ultime due modifiche appena illustrate. In questo modo, le notifiche sarebbero gestite automaticamente ed in maniera integrata dalle funzionalità conosciute del calendario Android.

PREZZI

Può essere utile associare, insieme alla lista della spesa, il prezzo di ogni prodotto che viene acquistato con l'obiettivo di tenere sotto controllo le spese e fare economia domestica. I prezzi sono memorizzati nell'applicazione per poi permetterne di visualizzare le spese attraverso grafici o istogrammi.

DIETA

Un'altra implementazione serve a tenere sempre sotto controllo la propria dieta alimentare, associando ad ogni ricetta o prodotto le informazioni nutrizionali quali numero di calorie, percentuale di carboidrati o simili. Probabilmente un comportamento di questo tipo prevede tanti di quegli aspetti che può essere più conveniente creare una nuova applicazione dedicata che si occupa solo di questo e che sfrutti l'eventuale content provider di Kulinaria appositamente realizzato.