

# **REPORT PROGETTO**

## **Laboratorio Applicazioni Mobili**

Martino Simonetti  
838295



Questo Report è volto a documentare l'applicazione mobile per applicativi Android MyHM.

Nello specifico si andranno ad analizzare i seguenti punti:

1. L'applicazione (inquadramento generale)
2. Progettazione
3. Implementazione della struttura
4. Interfaccia grafica
5. Testing

## 1. L'applicazione

MyHM è un applicazione Android che permette all'utente di registrare e monitorare nel tempo il proprio *peso*, la propria *temperatura corporea* e la propria *glicemia*.

Creando un nuovo *report* l'utente potrà inserire questi dati all'interno dell'app e potrà filtrarli su base giornaliera oppure a seconda dei valori inseriti. Sarà inoltre possibile impostare la *priorità* (in un range da 1 a 5) che l'utilizzatore deciderà di dare a ciascun parametro.

L'utente potrà visualizzare graficamente i propri valori attraverso due tipologie di grafico (scatterplot e istogramma).

L'app prevede l'invio di notifiche per ricordare l'inserimento di un report giornaliero. La notifica arriverà all'orario richiesto e potrà essere richiesto l'invio della stessa 5, 10 o 20 minuti più tardi.

## 2. Progettazione

Per lo sviluppo dell'app, scritta attraverso l'IDE Android Studio, è stato scelto il linguaggio Java.

MyHM è sviluppata con AndroidX ed è retrocompatibile fino alla versione di Android Marshmallow 6.0 (API 23).

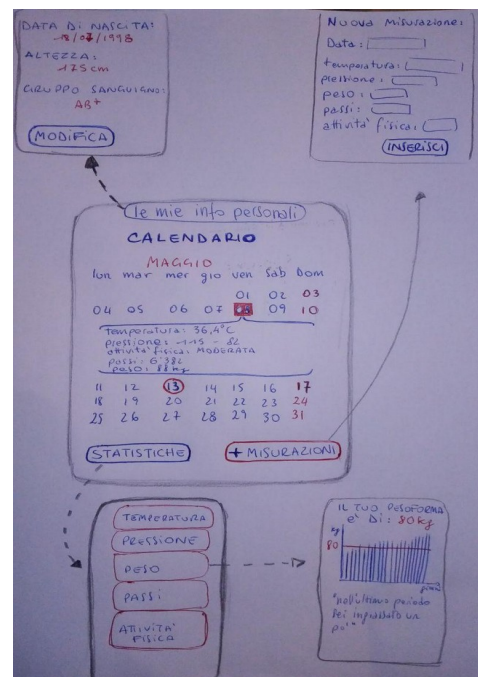
### 2.1 Struttura

In fase di progettazione sono state valutate molteplici ipotesi sulla possibile struttura della UI.

Nell'immagine a fianco è rappresentata un'ipotesi di struttura poi scartata perché valutata meno immediata di quella scelta.

MyHM è strutturata attraverso una tabView attraverso la quale si possono raggiungere le quattro principali schermate dell'app:

1. ME
2. CALENDAR
3. NEW REP
4. GRAFICI.



Da queste quattro principali app è possibile raggiungere altre schermate dell'app. La scelta della tabView è dovuta principalmente alla preferenza personale dello sviluppatore e a supportare la facilità d'utilizzo.

### 2.2 Salvataggio dei dati

Per il salvataggio dei dati all'interno dell'applicazione sono stati utilizzati il database Room e le SharedPreferences.

#### 2.2.1 Room

Per utilizzare e importare il DB è stato necessario seguire la seguente guida: <https://developer.android.com/training/data-storage/room>.

Nel DB creato con Room sono stati salvati esclusivamente i dati inseriti nei report.

I file coinvolti nel salvataggio dei dati nel DB sono i seguenti:

- **Reports.java**  
Le istanze di questa classe saranno inserite nel DB. Ogni oggetto reports avrà i seguenti attributi: id (chiave primaria), Peso, Temperatura, Glicemia, Note, Data (viene salvata come String, per scelta implementativa), Priorità (la priorità globale del report: media delle priorità di peso, temperatura e glicemia). Reports.java utilizza Tupla.java come classe di “supporto”: i dati relativi alla salute sono di tipo Tupla, ciò permette di salvare sia il valore, sia la priorità associata al singolo dato.  
Tutti i dati sono salvati in unica tabella chiamata “datiDB”.
- **AppDatabase.java**  
L’istanza *appDatabase* di AppDatabase.java è l’oggetto del DB.
- **DataAccessObject.java**  
È l’interfaccia che permette di inserire, rimuovere, aggiornare gli elementi del DB.  
Nell’interfaccia sono inoltre definite le Query al DB.

I metodi definiti nell’interfaccia per l’accesso al DB sono richiamati attraverso delle classi che estendono AsyncTask.

L’utilizzo di chiamate asincrone è fondamentale per garantire la fluidità dell’app.

### 2.2.2 SharedPreferences

Attraverso le shared preferences è possibile salvare delle piccole variabili che non vengono cancellate dopo la chiusura né dell’app né del telefono. Nel progetto sono state utilizzate per salvare:

- il nome dell’utente,
- l’orario di invio delle notifiche,
- i valori delle priorità di peso, temperatura e glicemia,
- la variabile booleana che permette di sapere se si sta effettuando il primo accesso o un accesso successivo.

### 3. Implementazione della struttura

Come accennato in precedenza la parte principale dell'applicazione è una *tab*View che viene implementata attraverso una sola *Activity* (*MainActivity.java*) sulla quale attraverso le classi che implementano il framework *ViewModel* vengono caricati i differenti *Fragment*. Questa implementazione viene generata di default da Android Studio. Per adattarla alle esigenze del progetto è stato necessario modificare il *SectionsPagerAdapter* implementando degli switch case che permettono di inserire i nomi dei vari tab e creare le nuove istanze dei fragment per ogni sezione.

```
@Nullable
@Override
public CharSequence getPageTitle(int position) {
    switch (position) {
        case 0:
            return "ME";
        case 1:
            return "CALENDAR";
        case 2:
            return "NEW REP";
        case 3:
            return "GRAFICI";
    }
    return null;
}
```

```
@Override
public Fragment getItem(int position) {
    Fragment fragment = null;
    switch (position) {
        case 0:
            fragment = new MyData();
            break;
        case 1:
            fragment = new Home();
            // fragment = new Calendario();
            break;
        case 2:
            fragment = new NewReport();
            break;
        case 3:
            fragment = new Grafici();
            break;
    }
    return fragment;
}
```

Sono presenti altre Activities:

- StartActivity

Viene visualizzata solo al primo accesso dell'app.

L'intent per la sua apertura si trova nel metodo *“onCreate”* della *MainActivity*. Per garantire ciò è stata

```
SharedPreferences prefs = getSharedPreferences( name: "prefs", MODE_PRIVATE);
boolean firstStart = prefs.getBoolean( s: "firstStart", b: true);

if (firstStart){
    startActivity(new Intent( packageContext: MainActivity.this, StartActivity.class));
    SharedPreferences.Editor editor = prefs.edit();
    editor.putBoolean( s: "firstStart", b: false);
    editor.apply();
}
```

creata la variabile *firstStart* inizialmente settata a TRUE verrà poi settata a FALSE durante l'esecuzione dell'activity stessa e salvata nelle shared preferences impedendo successivi accessi all'activity.

- Activity per i grafici

- PostponiActivity

Questa Activity viene richiamata soltanto premendo il tasto *“POSTPONP”* presente nelle notifiche.

L'intent che la richiama si trova nel *NotificationBroadcastReciver*.

- ReportsActivity

## 4. Interfaccia grafica

L'interfaccia, nelle diverse schermate, è composta da numerosi elementi disposti all'interno di un Layout; in particolare è stato utilizzato con prevalenza il ConstraintLayout per la sua ottima maneggevolezza attraverso l'interfaccia grafica di Android Studio.

### 4.1 Costruzione della UI

Sono stati utilizzati elementi differenti:

- Bottoni,
- Widgets,
- Aree di testo,
- Contenitori.

#### 4.1.1 CalendarView

La visualizzazione del calendario è stata costruita utilizzando il widget di default di Android.

All'interno della classe *Home* che si occupa della “gestione” degli elementi presenti nel *fragment\_calendario.xml*, è stato implementato l'evento che accadrà al tocco su un giorno del CalendarView.

```
calendario = view.findViewById(R.id.calendarView);

calendario.setOnDateChangeListener((view, year, month, day) -> {
    month = month + 1;

    date = month + "/" + day + "/" + year;
    // Toast.makeText(view.getContext(), date, Toast.LENGTH_LONG).show();
    new AsyncTaskRiceviReports().execute();
});
```



Il metodo *onSelectedDayChange* setta la variabile globale *data* alla data selezionata nel calendario, subito dopo viene effettuata la chiamata al DB dal quale vengono

estratti tutti i report all'interno di una lista, i report con data uguale a quella della variabile globale *data* vengono salvati in un'altra lista che verrà passata all'*adapter* della *recycleView*.

#### 4.1.2 RecycleView

Le *RecycleView* vengono utilizzate sia per la visualizzazione dei report sotto al calendario, sia per la visualizzazione di tutti i report con possibilità per il filtraggio.

Le *recycleView* sono costruite a partire dal file *fragment\_2recycle.xml* che definisce il layout della 'scheda' che verrà riprodotta molteplici volte.

Peso	Temperatura	Glicemia
Line 2	Line 4	Line 6
Note	Line 8	

Il compito di ricevere la lista dei report e generare tanti *fragment\_2recycle* quanti sono gli elementi della lista è svolta dal *RecycleAdapter.java*. Attraverso i getter degli oggetti *Reports* vengono recuperati i dati e l'*adapter* sostituisce i valori effettivi alle *TextView* "Line2", "Line4", "Line6" e "Line8".

##### 4.1.2.1 Le gestures sulle recycleView

Sono state implementate delle gestures applicabili ad ogni cella.

```
ItemTouchHelper.SimpleCallback simpleCallback = new ItemTouchHelper.SimpleCallback(0, swipeDirs: ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT){
    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder target) {
        return false;
    }
    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
        int position = viewHolder.getAdapterPosition();
        switch (direction){
            case ItemTouchHelper.LEFT:
                if(!exampleList.get(position).getNote().equals("Report riassuntivo")){
                    new AsyncTaskEliminaReport().execute(exampleList.get(position));
                }
                exampleList.remove(position);
                mAdapter.notifyItemRemoved(position);
                break;
            case ItemTouchHelper.RIGHT:
                openDialog(exampleList.get(position));
                new AsyncTaskRiceviReports().execute();
                break;
        }
    }
}
```

##### Swipe verso sinistra

Permette di eliminare l'elemento.

##### Swipe verso destra

Permette di modificare il report, si aprirà quindi un *dialog* che permetterà di inserire i nuovi valori.

Peso	Temperatura	Glicemia
98.0	37.0	80.0
Note	Primo rep	
	Peso	Temperatura
	90.0	36.0
	Note	Secondo rep
Peso	Temperatura	Glicemia
98.0	37.8	78.0
Note		

Peso	Temperatura	Glicemia
98.0	37.0	80.0
Note	Primo rep	
Temperatura	Glicemia	
0	80.0	
ondo rep		
Peso	Temperatura	Glicemia
98.0	37.8	78.0
Note		

Solo quando la `recyclerView` è presente indipendentemente dal calendario si rende necessario avere altre due gestures.

#### **Tocco sull'item**

Al tocco viene mostrato un *Toast* che mostra a display la data relativa al report.

#### **Scroll verso il basso sull'intera recyclerView**

Mentre nella sezione “CALENDAR” per ricaricare le ‘schede’ è sufficiente un click sul giorno del calendario, nella sezione “I MIEI REPORT” si rende necessario inserire la `recyclerView` in una `scrollView` che permette di aggiornare i report.

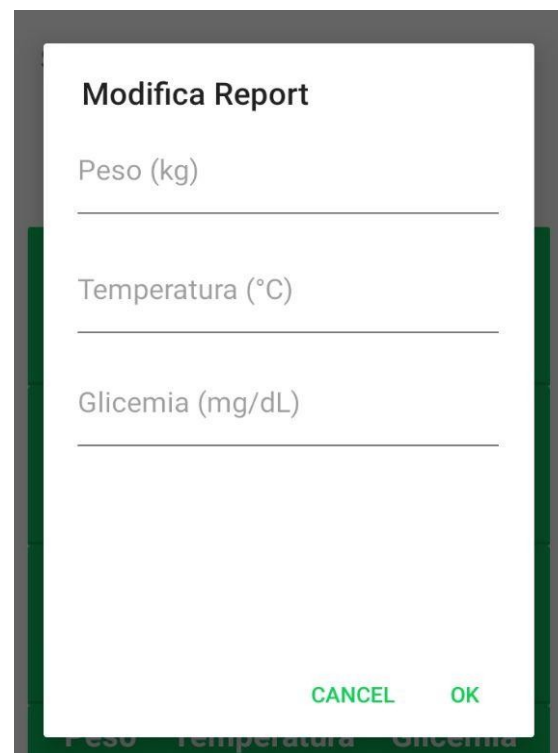
#### **4.1.3 Dialog**

In seguito allo swipe verso destra verrà aperto una dialog window che si occuperà della modifica del repo, fornendo una maschera per l'inserimento dei nuovi dati. Al click sul pulsante di conferma la modifica viene effettuata sul DB tramite chiamata asincrona.

Di tutto ciò è gestito dal file `DialogModficaRep.java` che estende `AppCompatActivity`.

Per rendere possibile alla classe di dialogare col DB è stato necessario modificare il costruttore.

```
public DialogModficaRep(Reports report) { this.reportNew = report; }
```



#### **4.1.4 Notifiche**

Le notifiche vengono create dalla classe `NotificationBroadcastReciver` solo se per il giorno corrente non sono già stati inseriti dei repo (si esegue un controllo sul DB).

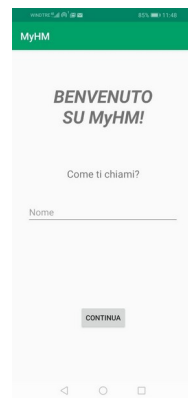


Il `NotificationBroadcastReciver` viene attivato dall'`AlarmManager` nella `MainActivity`, nella `PostponiActivity` e nel fragment `MyData`.

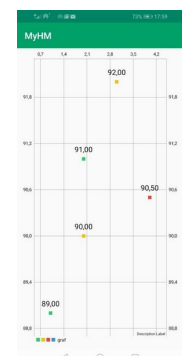
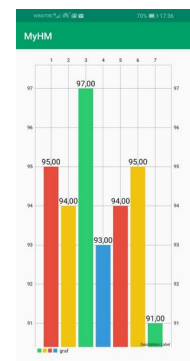
La costruzione della notifica prevede che si creino due `PendingIntent` differenti che permetteranno al tocco sui due pulsanti l'apertura di due activity distinte.



## 4.2 Mappa



(Al primo accesso)



## 5. Testing

L'app è stata testata principalmente su due telefoni emulati:

- Nexus 4 API 23,
- Pixel 2 XL API 23

telefoni fisici:

- Huawei Y6 2019 Android 9
- One Plus 5 Android 10
- Huawei Mate 10 Pro Android 10

In fase di testing sono emerse alcune piccole problematiche per quanto riguarda il layout di alcuni fragment.

È inoltre emerso l'esigenza di “spiegare all'utente l'utilizzo di alcune gestures.