ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

PERSONAL HEALTH MONITOR

PROGETTO DI LABORATORIO DI APPLICAZIONI MOBILI 2019/2020 DI MARTINA SOSTO 0000830768

REQUISITI E SPECIFICHE FUNZIONALI

Nel seguente progetto lo studente è tenuto a implementare un'applicazione interattiva per tenere traccia delle informazioni personali sulla propria salute da salvare in specifici report giornalieri. In particolare, l'applicazione dovrebbe essere in grado di gestire i report all'interno di un calendario, inviare notifiche e tracciare i dati secondo le specifiche.

CREAZIONE E MODIFICA DI REPORT

L'applicazione deve essere in grado di creare, modificare ed eliminare i rapporti sulla salute, questi sono riassunti in delle informazioni tracciati dall'utente, i report devono essere salvati ogni volta che l'utente ritiene che sia una buona idea e almeno una volta al giorno.
Ogni rapporto deve includere un numero minimo di due informazioni relative alla salute dell'utente (ad es. Temperatura corporea, pressione sanguigna, indice glicemico, ecc.).
Ogni informazione ha un'importanza, cioè un indice che specifica il livello di attenzione che richiede tale parametro (da 1 a 5) e ogni rapporto ha una nota opzionale che può essere integrata con informazioni ausiliarie. I report vengono archiviati dall'applicazione (si consiglia vivamente di utilizzare un database) e deve esserci la possibilità di mostrare i report su base giornaliera, come ad esempio all'interno di un calendario. Nel caso di più report per lo stesso giorno, è necessario creare un report di riepilogo, in cui le informazioni sanitarie sono la media di tutti i dati raccolti di quel giorno.
Ci deve essere anche la possibilità di visualizzare i rapporti secondo alcuni filtri (ad esempio, solo i rapporti con importanza impostata su 5).

NOTIFICHE

L'applicazione deve avvisare l'utente se non ha ancora inserito un rapporto giornaliero. In questo caso, l'utente può eseguire le seguenti azioni: rinviare il promemoria (in tal caso all'utente verrà richiesta un'altra ora e data dello stesso giorno) o aprire direttamente dall'interno della notifica il modulo per la gestione del rapporto. Il tempo in cui la notifica viene inviata dall'applicazione può essere impostato dall'utente da una pagina delle impostazioni. L'applicazione deve inoltre informare l'utente se la media dei dati raccolti per un'informazione - di importanza superiore a 3 - in un periodo di tempo prefissato ha superato una soglia prefissata. L'utente può personalizzare i parametri precedenti da una pagina delle impostazioni, ovvero può decidere quali informazioni devono essere monitorate, per quanto tempo e quale soglia non deve essere raggiunta.

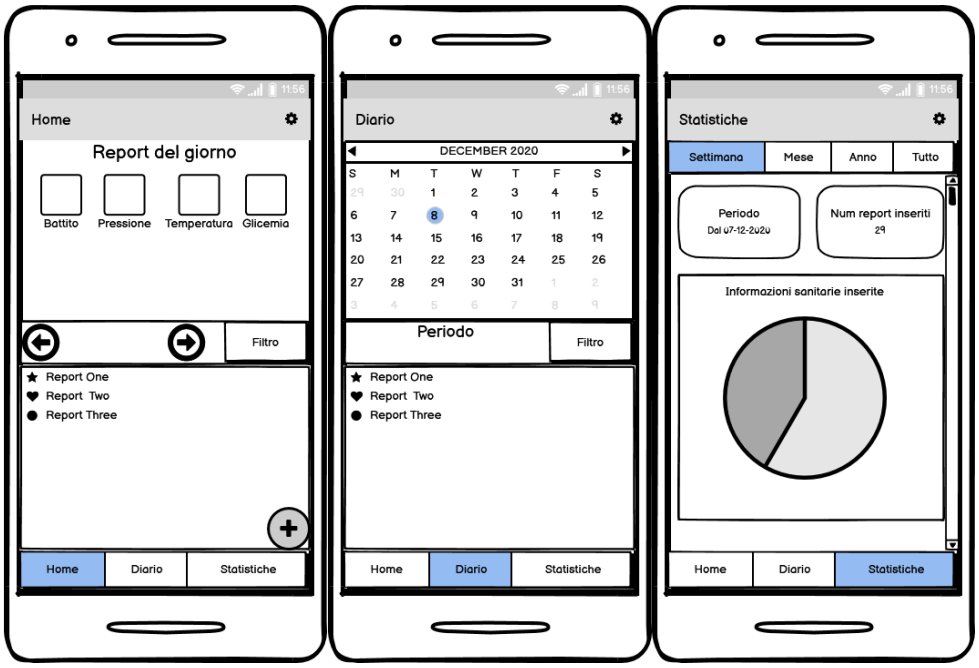
GRAFICI

L'applicazione deve essere in grado di raccogliere statistiche sull'utilizzo che visualizzano almeno due grafici di qualsiasi tipo (grafico a torta, diagramma a riquadri, istogramma, diagramma a linee, ecc.) che mostrano dati utili (ad

esempio la variazione di informazioni sanitarie nell'arco di una settimana, la variazione del numero di rapporti raccolti ogni giorno, ecc.).

SCHERMATE E FUNZIONALITÀ

L'applicazione comprende tre schermate principali e diverse schermate secondarie, che compariranno a seconda delle funzionalità ricercate. Le schermate principali sono la Home, il Diario e le Statistiche, queste hanno dei contenuti e delle funzionalità diverse ma tutte e tre sono incorniciate da un layout verde a tema chiaro e a sfondo scuro, che permette di mostrare i titoli, accedere ad altre schermate principali ed entrare nella schermata delle Impostazioni descritta più avanti.



HOME

La home ha la funzione di mostrare e riassumere i report di una singola giornata e si divide in due sezioni distinte. Nella parte alta verrà visualizzato il giorno in analisi e la media dei valori dei report aggiunti in tale data.

Figura 2. Home in landscape mode

Al centro ci sono le frecce di navigazione tra i vari giorni e il bottone del filtro. Cliccando sulla freccia di sinistra verrà visualizzato il giorno precedente e cliccando sulla freccia di destra, il giorno successivo. Abbiamo la possibilità di spostarci tra i giorni anche trascinando il dito (swipe) da destra verso sinistra sulla lista dei report per andare al giorno precedente, e da sinistra verso destra per andare al giorno successivo. Cliccando sul filtro comparirà il popup di selezione dei valori del filtro, che esamineremo nella sezione "Filtro".

Nel resto della pagina viene visualizzata la lista dei report aggiunti durante il giorno in considerazione in ordine cronologico inverso (ovvero dal più al meno recente). Affronteremo la visualizzazione dei report nella sezione "Report".

Figura 1. Schermata Home in portrait mode

Se nel giorno selezionato non sono stati ancora aggiunti report, allora la struttura della parte alta rimarrà invariata, ma non verranno visualizzate le medie dei valori e la lista risulterà vuota, al suo posto comparirà un avviso che notifica l'assenza di dati.

In basso a destra (se il dispositivo è in verticale – Figura 1) o in alto a sinistra (se il dispositivo è in orizzontale – Figura 2) sarà presente un bottone rotondo con un "+" al di sopra, toccandolo si aprirà una nuova visuale che permetterà di aggiungere un nuovo report, esamineremo questa schermata nella sezione "Nuovo report e Modifica report". I report che vengono aggiunti in questa schermata verranno salvati alla data odierna qualsiasi sia la data che stiamo visualizzando sulla Home.

DIARIO

Anche il Diario, come la Home è divisa in due parti [guardare la Figura 3 per la portrait mode e la figura 4 per la landscape mode]. Nella parte alta viene visualizzato il calendario dei report, che possiamo sfogliare e cliccare, questo indica il giorno odierno colorando di rosso il numero del giorno attuale e marcando con un pallino rosso le giornate contenenti almeno un report.

Sotto il calendario abbiamo il bottone del Filtro e un'etichetta che indica quale periodo o giorno stiamo considerando per la visualizzazione dei report.

Infatti, nella parte restante della pagina abbiamo una lista di report, questa inizialmente è popolata dai report del mese corrente, ma sfogliando il calendario tramite le apposite frecce ai lati dell'intestazione del mese, avremo la possibilità di visualizzare i report dei mesi precedenti. La navigazione tra i mesi avviene anche facendo swipe verso destra o sinistra sulla lista dei report.

Cliccando una data segnalata da un putino rosso avremo la possibilità di visualizzare tutti i report aggiunti in tale data. Se nel mese considerato non ci sono report comparirà un avviso come nella schermata Home, che indica che in tale periodo non ci sono dati, la stessa cosa accade cliccando su una data senza il marker rosso.

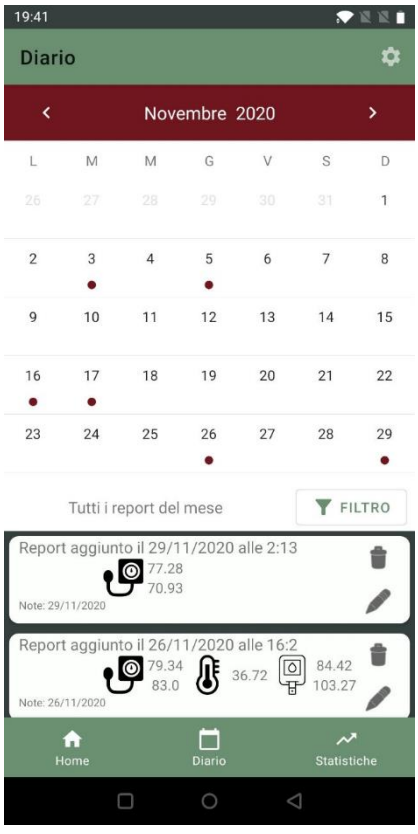


Figura 3. Diario in portrait mode

Figura 4. Diario in landscape mode scroll 1

Figura 4. Diario in landscape mode scroll 2

STATISTICHE

La pagina delle statistiche non è sezionata in due come le due schermate precedenti, questa è una lista di schede contenenti grafici e sondaggi inerenti ai report aggiunti precedentemente [Figura 5 per la landscape mode e figura 6 per la portrait mode].

Figura 5. Statistiche in landscape mode scroll 1

In alto abbiamo quattro bottoni di navigazione tra periodi, infatti questi permettono di esplorare un periodo specifico, questo può essere una Settimana, Mese, Anno o Tutto.

Subito sotto i bottoni troviamo due schede che indicano il periodo indicato [ad esempio, se il giorno attuale è il 07/12/2020, allora cliccando sul bottone con scritto Settimana, comparirà Periodo: Dal 07/12/2020 al 13/12/202] e il numero di report aggiunti in tale periodo.

In base alle informazioni registrate, il resto della schermata verrà modificata: se avremo dei report nel periodo indicato, allora i dati di tali report aggiorneranno i grafici con i valori contenuti al loro interno, altrimenti un avviso ci informerà che non ci sono report nel periodo da noi selezionato.

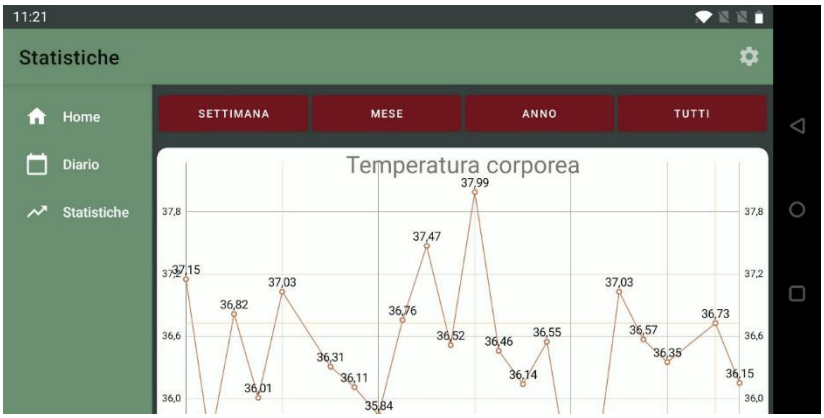


Figura 5. Statistiche in landscape mode scroll 2

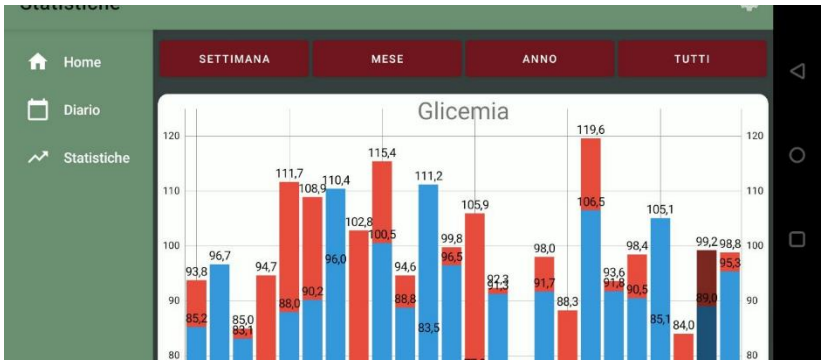


Figura 5. Statistiche in landscape mode scroll 3

I grafici che potremmo visualizzare sono in totale cinque. Il primo grafico è un aerogramma (grafico a torta), questo indica quali tipi di valori abbiamo registrato dai report da noi aggiunti. Per visualizzare questo grafico basterà aggiungere un report con un valore qualsiasi. Dal secondo grafico in poi verranno visualizzati dei diagrammi cartesiani e istogrammi (grafico a barre) che variano in base al valore dei dati indicati nei report.

Abbiamo un grafico per ogni tipologia di valore che possiamo inserire nel report. Consideriamo il grafico inerente ai valori del battito cardiaco. Per monitorare i dati registrati, ho usato un diagramma cartesiano dinamico, infatti, sull'asse delle ascisse verrà visualizzato in forma numerica il valore medio dei dati inseriti nei report che interessano il battito cardiaco e sull'asse delle ordinate verranno visualizzate le date che compongono tali periodi.

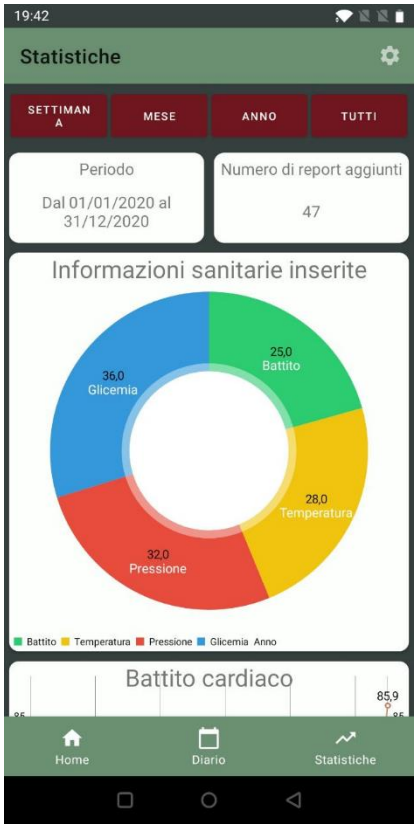


Figura 6. Statistiche in portrait mode scroll 1

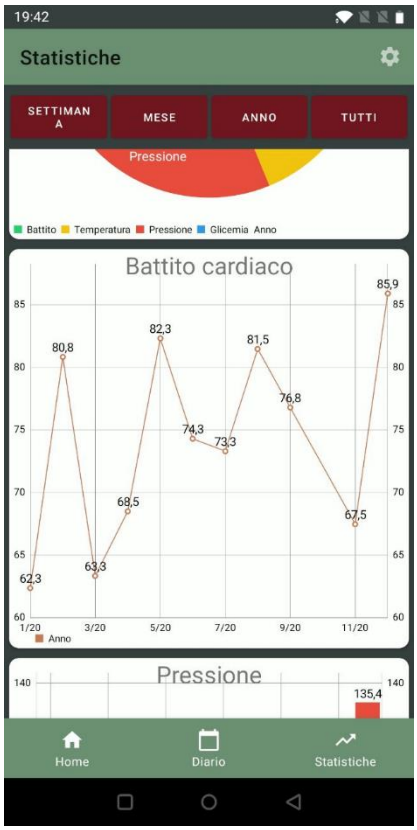


Figura 6. Statistiche in portrait mode scroll 2

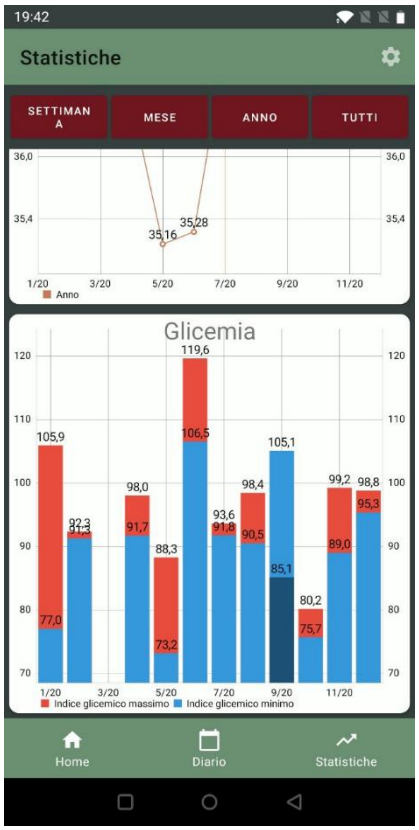


Figura 6. Statistiche in portrait mode scroll 3

I grafici inerenti ai valori della pressione e della glicemia vengono visualizzati con un istogramma con due barre sovrapposte, che mostra le differenze tra i due valori limite misurati. Se nel report inseriamo solo uno dei due valori, allora verrà visualizzata solo una barra delle due.

Come ho accennato precedentemente, i grafici sono dinamici, quindi meno dati vengono raccolti, meno dati mostreranno.

NUOVO REPORT E MODIFICA REPORT

Le schermate che consentono di inserire un nuovo report [Figura 8] o di modificare un report già esistente [Figura 7] non hanno visivamente differenze, meno che la schermata di modifica del report conterrà già i valori settati precedentemente alla creazione o alla scorsa modifica del report.

I valori preesistenti possono essere modificati come tutti gli altri, gli unici dati che non possono essere mai modificati, neanche alla creazione del report, sono la data e l'ora di creazione. Questo significa che, eliminando un vecchio report, questo non sarà più disponibile e non avremo in nessun modo la possibilità di aggiungere nuovamente un report nella data della sua rimozione.

Toccando il bottone in fondo alla schermata, potremo "salvare" i dati da noi inseriti e quindi tornare alla Home -se proveniamo dalla home e stiamo modificando o aggiungendo un report- oppure al diario -se stiamo modificando un report.

In questa ultima fase ci saranno dei controlli che servono a non inserire dei valori erronei nel database. I limiti superiori e inferiori dei vari valori sono descritti nella sezione "Caratteristiche dei report", superando tali soglie comparirà un messaggio di errore e non potremo salvare i dati e tronare alla schermata precedente; inoltre, possiamo salvare un report solo se inseriamo almeno due valori, la casella delle "note" non è compresa.

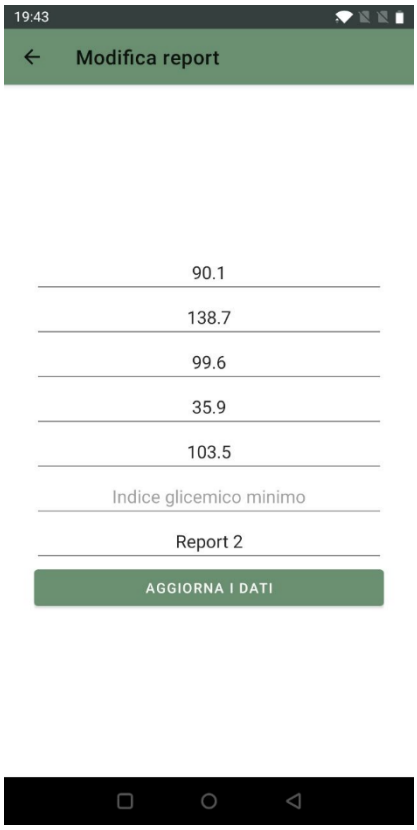


Figura 7. Modifica report

Figura 8. Nuovo report

IMPOSTAZIONI

Come spiegato precedentemente, è possibile accedere alle impostazioni da tutte e tre le schermate principali. Nelle impostazioni [Figura 9] abbiamo la possibilità di modificare le nostre preferenze inerenti alle notifiche e al filtraggio dei valori che troviamo relativamente più interessanti.

Tramite uno "switch" -inizialmente impostato come spento- abbiamo la possibilità di abilitare o disabilitare le notifiche giornaliere; abilitandole ci sarà consentito di impostare una preferenza per la ricezione della notifica.

Per ogni valore inseribile nel report abbiamo uno slider che ci permette di impostare la rilevanza che diamo al valore, questi sono inizialmente tutti impostati a 2. Quando facciamo scorrere lo slider fino al numero 3, comparirà una sezione che ci permette di impostare le notifiche di "allarme" di tale valore, infatti ci verrà chiesto di specificare un periodo e un numero limite, se la media dei valori del periodo indicato supererà il limite ci verrà inviata una notifica alle nove di mattina del giorno stesso, o del giorno seguente se già passata.

Anche qua, come quando inseriamo un nuovo report, abbiamo dei controlli sui valori soglia che vengono specificati e sulle date, ad esempio una data che funge da limite superiore non può essere precedente alla data che funge da limite superiore. Abbiamo, però, la possibilità di monitorare i dati di un singolo giorno selezionando come date limite il giorno stesso.

Figura 9. Impostazioni

FILTRO

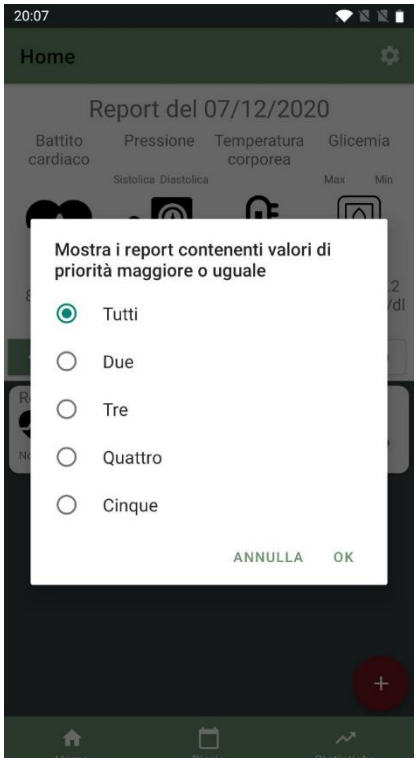


Figura 10. Filtro

Cliccando sul bottone Filtro viene visualizzato un popup [Figura 10] che permette di selezionare un numero tra 1 e 5, questo permette di indicare le nostre preferenze riguardanti le priorità dei report che vogliamo visualizzare. La priorità di un report è il numero di priorità più alto tra i suoi valori. Ad esempio, cliccando su "Tutti" visualizzeremo tutti i report, invece cliccando sul numero 5 verranno visualizzati solo report contenenti almeno un valore con priorità impostata a 5.

REPORT

I report vengono visualizzati solo tramite liste [Figura 11], e solo nella schermata Home e Diario. Ogni report contiene solo i valori che sono stati inseriti alla sua creazione o nella fase di modifica.

Possiamo modificare un report, cliccando sull'icona della matita o eliminarlo, cliccando sull'icona del cestino. Eliminando un report, comparirà un messaggio in basso che ci notifica che l'operazione di eliminazione è avvenuta con successo. Sul messaggio c'è un messaggio con scritto "Cancella operazione" [Figura 12], toccandolo annulleremo l'ultima operazione svolta.

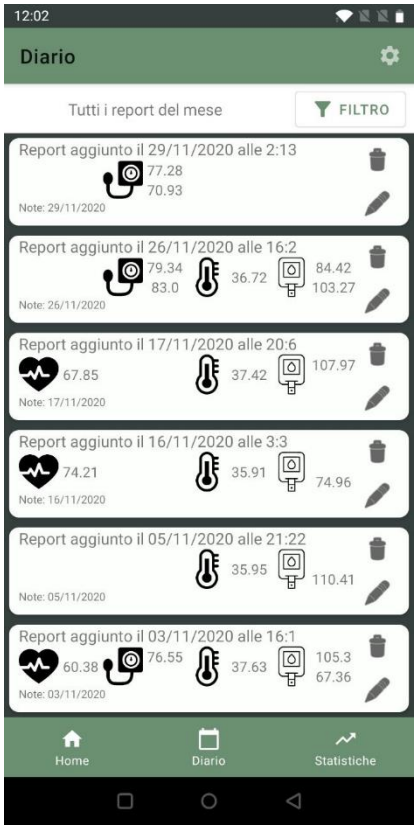


Figura 11. Lista di report del mese di novembre

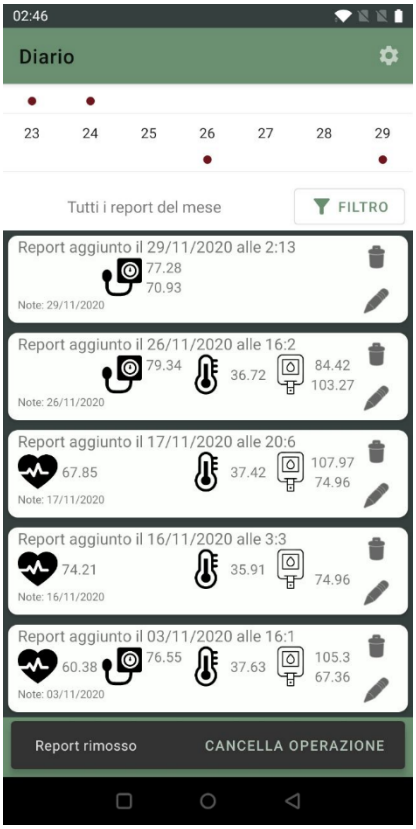


Figura 12. Avviso alla rimozione di un report

NOTIFICHE

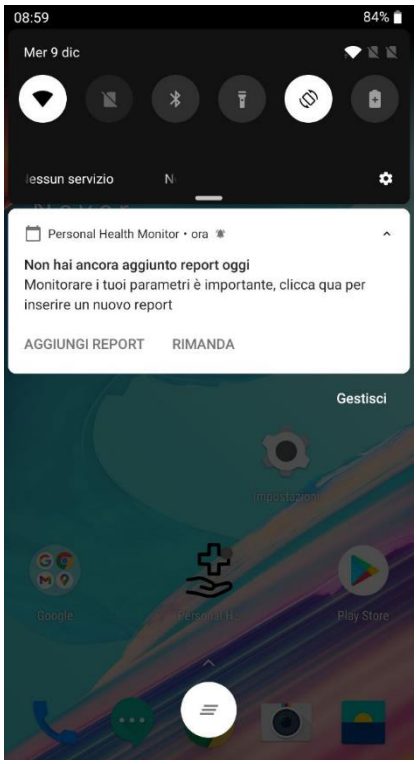


Figura 13. Notifica giornaliera

Abbiamo due tipi di notifiche, quelle giornaliere e quelle di "allerta".

Le notifiche giornaliere [Figura 13] vengono visualizzate quando la spunta inerente alle notifiche giornaliere che si trova nella schermata delle impostazioni è accesa, queste compariranno all'orario scelto dall'utente. Quando compare una notifica giornaliera abbiamo la possibilità di aggiungere un nuovo report oppure rimandare la notifica prossimamente.

Selezionando il primo caso si aprirà il format per inserire nuovi report [descritto nella sezione Nuovo report], altrimenti si aprirà una nuova schermata con all'interno un orologio [Figura 14], tramite questo avremo la possibilità di decidere a che orario vogliamo rimandare la notifica. La notifica può essere correttamente rimandata [Figura 15] o sarà possibile annullare l'operazione [Figura 16].

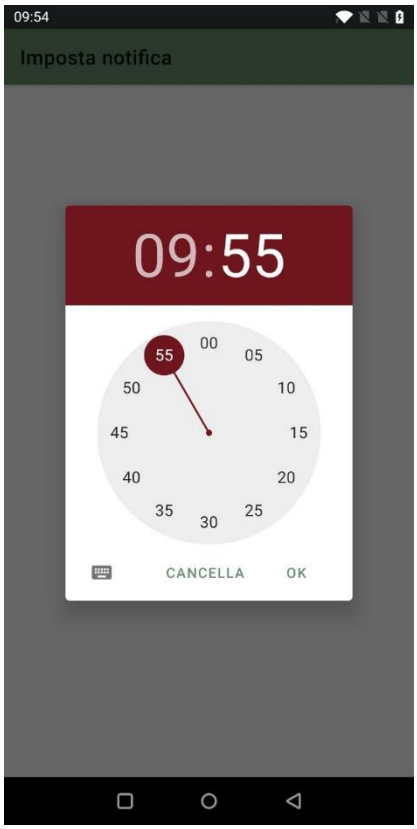


Figura 14. Selezionando “Rimanda”

Figura 15. Selezionando un orario di reinvio

Figura 16. Annullando l'operazione

Figura 17. Notifica di allerta

Le notifiche di “allerta” [Figura 17] compaiono tutti i giorni alle ore 09:00 ,se e solo se, l’utente ha impostato dei valori con priorità maggiore o uguale a 3 e almeno uno di questi supera il valore numerico limite nel periodo indicato.

Le notifiche di allerta vengono generate in base ai valori critici, per tanto, ogni volta che si cambiano le impostazioni dei valori limite avranno un contenuto diverso.

STRUTTURA DEL DATABASE

Diagramma ER

Reports	
id	int Primary Key
report_giorno	Date
report_ora	String
report_battito	double
report_temperatura	double
report_pressione_sistolica	double
report_pressione_diastolica	double
report_glicemia_max	double
report_glicemia_min	double
report_nota	String

Settings	
id	int PrimaryKey
settings_valore	String
settings_importanza	int
settings_inizio	Date
settings_fine	Date
settings_limite	double

Notifications	
id	int PrimaryKey
notification_status	boolean
notification_ora	int
notification_minuti	int

Per questo progetto ho utilizzato la libreria Room che si dedica alla persistenza su Sqlite caratterizzata da un approccio ORM (Object-Relational Mapping).

REPORTS

L'entità dei report serve a contenere, appunto, i dati inseriti dall'utente nei vari report.

- Id: è una chiave primaria intera che si auto incrementa e serve a tenere il conto dei reports inseriti in tabella
- Report_giorno: è un dato di tipo Date e non è mai Null, contiene la data di inserimento del report da parte dell'utente. In fase di ricerca, ho imparato che i database Room non permettono di salvare informazioni di tipo Date, per via ho marcato dei metodi che permettessero di fare il casting di valori Date in Long (e viceversa) con @TypeConverter, in questo modo il database è in grado di immagazzinare valori di tipo Date salvandoli in formato Long.
- Report_ora: è un dato di tipo String e non è mai Null, la stringa contiene la data in formato "h:mm" secondo il sistema orario delle 24 ore. Questo valore, come il giorno, viene memorizzato alla creazione del report e mai più modificato.
- Report_battito: contiene il valore di tipo double del battito cardiaco registrato nel report in questione, può essere un valore Null, ma non un valore double arbitrario, infatti, in base a ricerche online in fase di ricerca delle caratteristiche dei dati sanitari, sono venuta a conoscenza che i valori inerenti al battito cardiaco sono in genere:
    - inferiori a 60 battiti (al minuto): bradicardia

- tra 60 e 90-100 battiti al minuto: normale
- oltre i 100 battiti al minuto: tachicardia.

Per via di questo, ho implementato dei controlli che vengono effettuati all'inserimento e modifica del report.

- Report_temperatura, report_pressione, etc..: per quanto riguarda gli altri campi double della tabella, hanno le stesse caratteristiche del campo report_battito, con la differenza che per ogni valore ho ricercato quale fosse il range di esistenza più opportuno, attribuendo in questo modo le restrizioni corrette sull'inserimento dei dati nel seguente modo.

  Temperatura:

  - Da 35,5 °C a 37 °C

  Pressione arteriosa:

CLASSIFICAZIONE dei LIVELLI di PRESSIONE ARTERIOSA in mmHg		
Categoria	**Sistolica**	**Diastolica**
Ipotensione	<100	<60
Ottimale	<120	<80
Normale	120-129	80-84
Normale-alta	130-139	85-89
Ipertensione di grado 1	140-159	90-99
Ipertensione di grado 2	160-179	100-109
Ipertensione di grado 3	>180	>110
Ipertensione sistolica isolata	>140	<90

  Indice glicemico:

  - Normalità a digiuno = 60-110 mg/dL
  - Alterata glicemia a digiuno = 110-125 mg/dL
  - Ridotta tolleranza glicidica = 140-200 mg/dL
  - Diabete ≥ 126 mg/dL
- Report_nota: contiene il valore stringa di ulteriori dati della quale si vuole tenere traccia nel report.

SETTINGS

L'entità dei settings è molto diversa da quella dei reports, infatti di questo tipo abbiamo una sola entità, abbiamo solo una tabella di tipo settings che memorizza le impostazioni impostate dall'utente. L'entità stessa, insieme ai suoi campi, viene creata al primo avvio dell'applicazione, infatti viene creata una riga per ogni valore che possiamo avere nel report e l'importanza impostata di default a 2.

- Id: è una chiave primaria intera che si auto incrementa e serve a tenere il conto dei settings inseriti in tabella, il numero delle righe è fisso.
- Settings_valore: contiente una stringa che fa riferimento ai campi dell'entità report. Ad esempio se stiamo impostando le preferenze di importanza del battito cardiaco, ovvero il campo report_battito, questo campo conterrà la stringa "report_battito".
- Settings_importanza: contiene il valore intero che indica l'importanza del dato valore, questa viene di default impostata a 2 e può variare tra 1 e 5, se al di sopra di 3, verrà inviata una notifica di allarme se la media dei valori del periodo indicato supera la soglia limite.

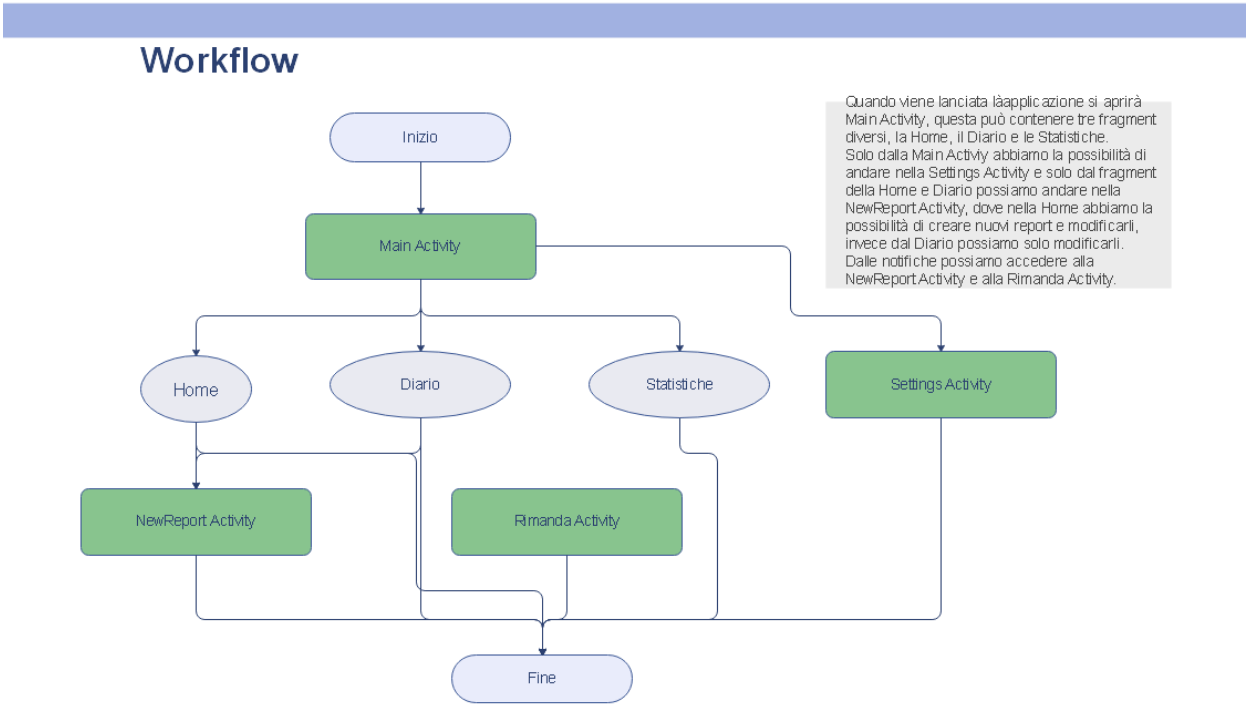
- Settings_inizio: contiene il formato Date del limite inferiore oltre alla quale dobbiamo monitorare il valore. Questo viene impostato nella schermata Impostazioni. Anche qua, come per il campo report_giorno, viene utilizzato un TypeConverter per poter memorizzare il valore di tipo Date.
- Settings_fine: contiene il formato Date del limite superiore sotto la quale dobbiamo monitorare il valore.
- Settings_limite: contiene il valore numerico double, oltre alla quale dobbiamo inviare la notifica di allarme nel caso la media dei valori nel periodo indicato venga superata.

NOTIFICATIONS

Come per i settings, la tabella è una sola e ha una sola riga e viene creata al primo avvio dell'applicazione. Il formato che usiamo per la memorizzazione dell'ora e dei minuti è quello delle 24 ore, ovvero hh:mm.

- Id: è una chiave primaria intera che si auto incrementa e serve a tenere il conto dei settings inseriti in tabella, il numero delle righe è fisso, ed è uguale a 1.
- Notification_status: è un valore booleano che descrive se le notifiche giornaliere sono o meno attive, inizialmente è impostato come FALSE, ovvero disattivate.
- Notification_ora: contiene il valore intero dell'ora in cui vogliamo che venga visualizzata la notifica giornaliera, inizialmente Nullo, modificabile dalla schermata delle Impostazioni.
- Notification_minuti: contiene il valore intero dei minuti in cui vogliamo che venga visualizzata la notifica giornaliera, inizialmente Nullo, modificabile dalla schermata delle Impostazioni.

DOCUMENTAZIONE TECNICA



MAIN ACTIVITY

Nella Main Activity ho implementato la configurazione dei vari menu, utilizzando una BottomNavigationView per la portrait mode e un NavigationView per la landscape mode. Vengono inoltre inizializzati i model che mi permettono di gestire i database e il valore del filtro, inizialmente impostato a 1, in questo modo abbiamo la possibilità di visualizzare ogni tipo di report.

Per avere una maggiore esperienza di testing, ho utilizzato le SharedPreferences per inizializzare le entità dei settings e notification e istanziare, inoltre, un numero arbitrario di reports (io ho scelto 100) in questo modo.

```
//PRIMO AVVIO
boolean firstRun = getSharedPreferences("preferences", MODE_PRIVATE).getBool-
ean("firstRun", true);
if(firstRun) {
    Utility.randomData(100);
    //startService(new Intent( this,Alarm.class));
    getSharedPreferences("preferences", MODE_PRIVATE).edit().putBoolean("firstRun",
false).apply();
}
startService(new Intent( this, Alarm.class));
```

Dove Utility.randomData è la funzione che mi genera i dati randomici e istanzia le entità del database. Uso la Main activity anche per far partire il servizio di allarme, gestito dalla classe Alarm.

HOME FRAGMENT

Alla creazione del Fragment Home, istanzio un valore dataSel di tipo MutableLiveData, in questo modo ho la possibilità di osservarlo durante tutto il ciclo di vita del fragment. Questo valore è importante perché contiene la data del giorno che stiamo visualizzando. Ogni volta che utilizzo i bottoni o lo swipe per cambiare giornata, il valore immagazzinato in dataSel, varierà e ad ogni sua variazione verranno richieste al database nuove informazioni riguardanti i report del giorno selezionato.

I dati vengono inoltre modificati quando si seleziona un valore diverso di priorità tramite il bottone del filtro, questo infatti, come ad ogni modifica di dataSel, chiama la funzione updateList() che prima controlla la tabella dei settings, ed in base ai valori impostati prende i report dal database nel modo seguente.

```
//AGGIORNA TUTTO
private void updateList() {
    //CAMBIA LA LISTA DEGLI ELEMENTI IN BASE AL FILTRO DEI SETTINGS
    settingsViewModel.getmAllSettingsFilter(filtro.getValue()).observe(getViewLifecy-
cleOwner(), new Observer<List<Settings>>() {
        @Override
        public void onChanged(List<Settings> settings) {
//PRIMA PRENDO I VALORI DAL FILTRO E POI IN BASE A QUELLO PRENDO I REPORT CHE MI INTE-
RESSANO
            reportViewModel.getFilterReports(dataSel.getValue(), null, settings).ob-
serve(getViewLifecycleOwner(), new Observer<List<Report>>() {
                @Override
                public void onChanged(List<Report> reports) {
                    new getAVG().execute();
                    reportListAdapter.setReports(reports);
                    if(reports.size() > 0) {
                        recyclerView.setVisibility(View.VISIBLE);
                        CardNoReport.setVisibility(View.GONE);
                    }
                    else CardNoReport.setVisibility(View.VISIBLE);
                }
            });
```

```
        }
    });
}
```

Nel fragment Home abbiamo anche la visualizzazione delle medie dei report giornalieri, per quello ho usato una classe getAVG che estende AsyncTask e richiede al database direttamente la media dei valori dei report giornalieri.

Toccando il bottone del filtro compare un Single Choiche Dialog, che permette di selezionare la priorità dei report che vogliamo visualizzare.

Toccando il Floating action button verrà lanciato un intent che permette di aprire la nuova activity NewReport Activity, che verrà utilizzata per registrare un nuovo report.

NEW REPORT ACTIVITY

Lo sviluppo della NewReport Activity è molto semplice, come ho descritto precedentemente, questa activity viene usata sia per aggiungere un nuovo report, sia per modificarne uno già creato precedentemente. La prima operazione che viene svolta è un controllo sul bundle passato assieme all'intent che ha permesso la creazione dell'activity, se questo è nullo allora stiamo creando un nuovo report, altrimenti nell'istanza del bundle avremo l'id del report che andremo a modificare. Nel secondo caso è necessario prendere tutti i dati contenuti in tale report e visualizzarli nelle Edit Text della schermata, in modo tale da far notare all'utente che quei dati sono già stati impostati ma sono modificabili.

Quando viene premuto il bottone che aggiornerà i campi del report modificato o creerà il nuovo report verranno anche effettuati dei controlli sui valori inseriti per evitare che ci siano valori fasulli.

REPORT LIST ADAPTER

La classe ReportListAdapter permette di creare e visualizzare una RecyclerView contenente i report richiesti. Tramite la classe ReportViewHolder assegno i valori del report alle TextView in modo dinamico, vale a dire, che verranno visualizzate solo le informazioni presenti nel singolo report. Se determinate informazioni non sono presenti, allora anche visivamente non verrà visualizzata nessuna icona e annesso valore.

Uso la funzione setListenters() per gestire le azioni di tocco sull'icona della matita e del cestino, nel primo caso usiamo un intent per aprire la NewReport Activity, allegando un bundle con l'id del report da modificare, nel secondo caso creiamo una nuova istanza della classe SnackbarUndo che esamineremo in seguito.

DIARIO FRAGMENT

Alla creazione del Fragment Diario istanzio il calendario CalendarView e la lista dei report con una RecyclerVIew, i dati al suo interno e gli eventi all'interno del calendario si aggiornano tramite la funzione updateList() che aggiungerà al calendario le icone rosse degli eventi in cui ho un report e aggiungerà alla lista i report del mese.

```
//AGGIORNA LA LISTA DEI REPORT
private void updateList(){
    //Log.i("UPDATE", "");
    TXVGiorno.setText(R.string.diario_label2);
    Calendar Cmese = calendarView.getCurrentPageDate();
    int ultimo = Cmese.getActualMaximum(Calendar.DAY_OF_MONTH);
    Calendar Fmese = calendarView.getCurrentPageDate();
    Fmese.set(Calendar.DAY_OF_MONTH, ultimo);

    //CAMBIA LA LISTA DEGLI ELEMENTI
    settingsViewModel.getmAllSettingsFilter(filtro.getValue()).observe(getViewLifecy-
cleOwner(), settings -> reportViewModel.getFilterReports(Cmese.getTime(), Fmese.get-
Time(), settings).observe(getViewLifecycleOwner(), reports -> {
        reportListAdapter.setReports(reports);
```

```
        if(reports.size() > 0) {
            recyclerView.setVisibility(View.VISIBLE);
            CardNoReport.setVisibility(View.INVISIBLE);

            List<EventDay> events = new ArrayList<>();
            for (int i = 0; i<reports.size(); i++){
                Calendar calendar = Calendar.getInstance();
                calendar.setTime(reports.get(i).getData());
                events.add(new EventDay(calendar, R.drawable.ic_reportevent));
                calendarView.setEvents(events);
            }
        }
        else{
            CardNoReport.setVisibility(View.VISIBLE);
            List<EventDay> events = new ArrayList<>();
            calendarView.setEvents(events);
        }
    }));

}
```

Per sfogliare i mesi ho aggiunto dei listener che osservano i tocchi sulle frecce laterali al calendario e in seguito aggiornano, sempre tramite la funzione updateList(), gli elementi della schermata. Le stesse funzioni vengono svolte facendo swipe sulla RecyclerView.

Cliccando su un giorno specifico, il calendario rimarrà invariato e verrà solamente aggiornata la lista dei report, mostrando solo quelli della giornata tenendo conto della priorità impostata nel filtro.

Le funzionalità del bottone del filtro sono le stesse implementate nel fragment Home.

STATISTICHE FRAGMENT

Alla creazione del Fragment delle statische istanzio la variabile “periodo”, questa è di tipo MutableLiveData e mi permette di aggiornare i grafici in base appunto al periodo in considerazione. Ogni volta che un bottone in alto, riguardante il periodo da mostrare viene toccato, la variabile periodo verrà impostata con le stringhe “Settimana”, “Mese”, “Anno”, “Tutto” in base al periodo scelto. Ogni qualvolta cambia il periodo, utilizzo delle funzioni dichiarate nella classe Utility che mi permettono di ottenere le giornate limite dei periodi richiesti, tramite queste posso richiedere al database i dati che mi servono e creare così i grafici.

I grafici si aggiornano con i dati richiesti tramite la chiamata di diverse funzioni asincrone (una per ogni tipo di grafico), queste mi permettono di non bloccare la UI e non fare richieste sul MainThread. Ecco un esempio di sottoclasse asincrona.

```
//CONTA IL UMERO DI VALORI PRESENTI NEI VARI REPORT E CREA IL PIE CHART
class PieDataAsyncTask extends AsyncTask<Void, Integer, ArrayList<PieEntry>>{

    @Override
    protected void onPostExecute(ArrayList<PieEntry> pieEntries) {
        super.onPostExecute(pieEntries);

        if(pieEntries.size() > 0) {
            PieDataSet pieDataSet = new PieDataSet(pieEntries, periodo.getValue());
            pieDataSet.setColors(ColorTemplate.MATERIAL_COLORS);
            pieDataSet.setValueTextSize(12);
            pieDataSet.setValueTextColor(Color.BLACK);
            PieData pieData = new PieData(pieDataSet);
```

```
            pieChart.setData(pieData);
            pieChart.setDescription(getDescription());
            pieChart.animateXY(1000, 1000);
            pieChart.invalidate();
        }
    }

    @Override
    protected ArrayList<PieEntry> doInBackground(Void... voids) {
        //TXVNumReport.setText(String.valueOf(reportViewModel.getCOUNTVal(null, null,
inizio, fine)));
        ArrayList<PieEntry> PiedataVals = new ArrayList<>();
        PiedataVals.add(new PieEntry(reportViewModel.getCOUNTVal(KEY_BATTITO,null, ini-
zio, fine), "Battito"));
        PiedataVals.add(new PieEntry( reportViewModel.getCOUNTVal(KEY_TEMPERATURA,
null, inizio, fine) , "Temperatura"));
        PiedataVals.add(new PieEntry(reportViewModel.getCOUNTVal(KEY_PRESSIONESIS,
KEY_PRESSIONEDIA, inizio, fine), "Pressione"));
        PiedataVals.add(new PieEntry(reportViewModel.getCOUNTVal(KEY_GLICEMIAMAX,
KEY_GLICEMIAMIN, inizio, fine), "Glicemia"));
       return PiedataVals;
    }
}
```

La funzione sopra descritta richiede al database il numero di volte in cui è stato inserito un tipo di dato sul totale dei report, in questo modo siamo in grado di costruire il PieChart, ovvero l'aerogramma che compare a inizio schermata se abbiamo almeno un report nel periodo scelto.

La classe seguente, invece, viene utilizzata per costruire i LineChart, ovvero i grafici che utilizzo per i valori del battito cardiaco e della temperatura. Per prima cosa controllo che periodo vogliamo visualizzare e in base a quello richiedo al database la media del valore dei dati: giornalieri, se il periodo indicato è la settimana o il mese e mensili se il periodo indicato è l'anno o tutti i report. In questo modo possiamo creare dei grafici che non siano troppo ampi o troppo ridotti rispetto al periodo selezionato. Una volta reperiti i dati, questi vengono memorizzato in un ArrayList di Entry, che permetterà nella fase di onPostExecute di "disegnare" a tutti gli effetti il grafico.

```
//CREA IL CHART DEL BATTITO E DELLA TEMPERATURA PRENDENDO I DATI IN BASE AL PERIODO NEL
DB
class LineDataAsyncTask extends AsyncTask<String, Integer, ArrayList<Entry>>{
    private final ArrayList<String> xAxisLabel = new ArrayList<>();
    private String valore;

    @Override
    protected void onPostExecute(ArrayList<Entry> entries) {
        super.onPostExecute(entries);

        LineDataSet lineDataSet = new LineDataSet(entries, periodo.getValue());
        lineDataSet.setValueTextSize(12);
        lineDataSet.setCircleColor(Color.rgb(197, 123, 87));
        lineDataSet.setColor(Color.rgb(197, 123, 87));
        lineDataSet.setValueTextColor(Color.BLACK);
        LineData lineData = new LineData(lineDataSet);

        if (valore.equals(KEY_BATTITO)){
            if(entries.size() > 0) {
                XAxis xAxis = battitoChart.getXAxis();
                xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
                xAxis.setDrawGridLines(true);
                xAxis.setGranularity(1);
                xAxis.setGranularityEnabled(true);
                battitoChart.setDescription(getDescription());
```

```
                battittoChart.animateXY(1000, 1000);
                battittoChart.setTouchEnabled(true);
                battittoChart.setPinchZoom(true);
                battittoChart.setData(lineData);
                battittoChart.setScaleEnabled(true);
                battittoChart.getXAxis().setValueFormatter(new IndexAxisValueFormat-
ter(xAxisLabel));
                battittoChart.invalidate();
            }
            else  root.findViewById(R.id.Cardbattito).setVisibility(View.GONE);
        }
        else{
            if(entries.size() > 0) {
                XAxis xAxis = temperaturaChart.getXAxis();
                xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
                xAxis.setDrawGridLines(true);
                xAxis.setGranularity(1);
                xAxis.setGranularityEnabled(true);
                temperaturaChart.setDescription(getDescription());
                temperaturaChart.animateXY(1000, 1000);
                temperaturaChart.setTouchEnabled(true);
                temperaturaChart.setPinchZoom(true);
                temperaturaChart.setData(lineData);
                temperaturaChart.setScaleEnabled(true);
                temperaturaChart.getXAxis().setValueFormatter(new IndexAxisValueFormat-
ter(xAxisLabel));
                temperaturaChart.invalidate();
            }
            else root.findViewById(R.id.Cardtemperatura).setVisibility(View.GONE);
        }
    }

    @Override
    protected ArrayList<Entry> doInBackground(String... strings) {
        valore = strings[0];
        ArrayList<Entry> dataVals = new ArrayList<>();
        Calendar Cinizio = Calendar.getInstance();
        Calendar Cfine = Calendar.getInstance();
        if(inizio == null && fine == null){
            Cinizio.setTime(Converters.StringToDate(SDF.format(reportViewModel.getMin-
MaxDateReport("MIN", valore))));
            Cfine.setTime(Converters.StringToDate(SDF.format(reportViewModel.getMinMax-
DateReport("MAX", valore))));
        }
        else {
            Cinizio.setTime(Converters.StringToDate(SDF.format(inizio)));
            Cfine.setTime(Converters.StringToDate(SDF.format(fine)));
        }
        Cfine.add(Calendar.DATE, 1);
        int c= 0;
        while(Converters.DateToLong(Cinizio.getTime())< Converters.DateTo-
Long(Cfine.getTime())){
            if(periodo.getValue().equals(KEY_ANNO) || periodo.getVa-
lue().equals(KEY_TUTTO)) {
                Calendar tmp = Calendar.getInstance();
                tmp.setTime(Cinizio.getTime());
                tmp.setTime(Utility.UltimoGiornoMese(tmp));
                Double value = reportViewModel.getAvgVal(valore, Cinizio.getTime(),
tmp.getTime());
                if(value != null){
                    dataVals.add(new Entry(c, value.floatValue()));
```

```
                }
                xAxisLabel.add((Cinizio.getTime().getMonth() + 1) + "/"+ (Cinizio.get-
Time().getYear() - 100));
                Cinizio.add(Calendar.MONTH, 1);
                Cinizio.setTime(Utility.PrimoGiornoMese(Cinizio));
            }
            else {
                Double value = reportViewModel.getAvgVal(valore, Cinizio.getTime(),
null);
                if (value != null) dataVals.add(new Entry(c, value.floatValue()));
                xAxisLabel.add(Cinizio.getTime().getDate() + "/"+ (Cinizio.get-
Time().getMonth() + 1));
                Cinizio.add(Calendar.DATE, 1);
            }
            c++;
        }
        return dataVals;
    }
}
```

Le stesse funzioni sono svolte dalla sottoclasse BarDataAsyncTask che utilizzo per disegnare i grafici dell'indice glicemico e della pressione, valori che necessitano di due dati numerici differenti.

SETTINGS ACTIVITY

La settings activity non è molto complessa, ma è piena di controlli, infatti per rendere il design più dinamico e meno compilativo, ho deciso di visualizzare inizialmente solo poche impostazioni ed è solo cambiando queste, compariranno in schermata ulteriori impostazioni da personalizzare.

Alla creazione dell'activity, richiedo al database gli attuali dati dei settings e delle notifiche così da poterle visualizzare. In seguito, osservo lo Switch delle notifiche giornaliere e le Seekbar di ogni valore per far comparire le impostazioni riguardanti l'orario di ricezione delle notifiche giornaliere e i valori limite fuori dai quali è necessaria una notifica.

Toccando il bottone del salvataggio, vengono effettuati dei controlli sulle impostazioni inserite, infatti le date dei valori devono essere consone (la data di inizio non può essere impostata dopo la data di fine – ma possono essere lo stesso giorno) e i valori limite sono simili a valori effettivamente monitorabili (ad esempio: non ha senso controllare che il battito cardiaco non super il valore 10, il battito cardiaco è normalmente tra 40 e 160 battiti al minuto).

ALARM

Ho ritenuto opportuno che la classe Alarm estendesse Service, questo perché una classe di questo tipo non ha bisogno di una visualizzazione grafica, ma deve svolgere funzioni di background da quando si apre l'applicazione fino al suo termine.

Come prima cosa, viene creato un canale delle notifiche e in secondo piano osserviamo costantemente le impostazioni delle notifiche e i report. Tramite il primo Observer abbiamo la possibilità di capire se l'utente è interessato a ricevere delle notifiche giornaliere o meno e in base a quello impostare un allarme, solo dopo aver controllato se nella giornata attuale è stato o meno già inserito un report. In questo modo possiamo gestire le notifiche giornaliere.

Per gestire le notifiche di allarme dobbiamo invece fare più controlli, è necessario richiedere al database i valori di settings per capire quali valori (che hanno una priorità maggiore o uguale a 3) osservare, una volta reperiti i dati dobbiamo controllare che nel periodo richiesto, non superino il limite indicato dall'utente, per fare ciò mandiamo un'altra richiesta al database che restituirà direttamente la media dei valori di quel periodo, dopo di che con un semplice controllo siamo in grado di capire cosa notificare all'utente, creiamo le stringhe di allarme, settiamo l'allarme e passiamo

le stringhe come extra dell'intent che verrà inviato al Broadcast Receiver che si occuperà di creare la notifica e visualizzarla. Mostro di seguito la funzione appena descritta.

```
    //PRENDO I VALORI MEDI NEL PERIODO INDICATO E, SE SOPRA IL LIMITE, AGGIUNGO LE
STRIGHE DI ALLARME A WARNING STRING
    public class AVGSettings extends AsyncTask<Settings, Void, String>{
        @Override
        protected void onPostExecute(String aString) {
            super.onPostExecute(aString);
            Log.i("WARNING STRING FINAL", aString);
            if(!aString.matches("")){
                //Log.i("WARNING STRING NOTIFICA", WarningString);
                Calendar alarmClock = Calendar.getInstance();
                alarmClock.set(Calendar.HOUR_OF_DAY, 9);
                alarmClock.set(Calendar.MINUTE, 0);
                alarmClock.set(Calendar.SECOND, 0);
                alarmClock.add(Calendar.DATE, 1);

                //Setto le notifiche di warning
                AlarmManager alarmManagerReportWarning = (AlarmManager) getSystem-
Service(ALARM_SERVICE);
                Intent intentWarning = new Intent(getApplicationContext(), Notifica-
tion_receiver.class);
                intentWarning.setAction(KEY_WARNING);
                intentWarning.putExtra("WarningString", aString);
                PendingIntent pendingIntentReportWarning = PendingIntent.getBroad-
cast(getApplicationContext(), NOTIFICATION_ID, intentWarning, PendingIntent.FLAG_UP-
DATE_CURRENT);

                Log.i("WARNING ALARM", Converters.DateToString(alarmClock.getTime())+ "
alle "+ alarmClock.getTime().getHours()+":"+ alarmClock.getTime().getMinutes());
                alarmManagerReportWarning.setRepeating(android.app.AlarmMan-
ager.RTC_WAKEUP, alarmClock.getTimeInMillis(), AlarmManager.INTERVAL_DAY, pendingInten-
tReportWarning);
            }
        }
//
        @Override
        protected String doInBackground(Settings... settings) {
            String warning ="";
            for (Settings mSettings : settings) {
                Double avg = reportViewModel.getAvgVal(mSettings.getValore(), mSet-
tings.getInizio(), mSettings.getFine());
                if (avg != null && avg > mSettings.getLimite()) {
                    warning += Utility.KeyToPrompt(mSettings.getValore()) + ": " +
tronca(avg) + "/" + mSettings.getLimite() + " tra il " + Converters.DateToString(mSet-
tings.getInizio()) + " e il " + Converters.DateToString(mSettings.getFine()) + "\n";
                    //Log.i("WARNING STRING", WarningString);
                }
            }
            return warning;
        }
    }
```

NOTIFICATION_RECEIVER

La classe Notification_receiver estende BroadcastReceiver, questa infatti viene utilizzata per catturare gli intent mandati dagli AlarmManager impostati nella classe Alarm e ostrare le notifiche.

Come prima cosa il Notification_receiver controlla l'azione che porta con sé l'intent che l'ha attivato e percepisce quale tipo di notifica vogliamo mostrare. Per le notifiche giornaliere e le notifiche giornaliere rimandate il funzionamento è statico ed è pressoché identico, viene creata una notifica utilizzando dei valori statici e visualizzata.

Per quanto riguarda, invece, le notifiche di allarme, anche qua la notifica viene costruita e visualizzata in modo statico, ma il contenuto al suo interno viene reperito come extra dall'intent ricevuto, in questo modo possiamo far leggere all'utente quali tipi di valore sono critici.

RIMANDA ACTIVITY

L'activity Rimanda può essere creata solamente toccando il bottone "Rimanda" sulla notifica giornaliera. Questa activity mostra semplicemente un TimePickerDialog, tramite questo l'utente può selezionare a quale ora vuole visualizzare nuovamente la notifica. Se settato correttamente, verrà istanziato un AlarmManager che manderà l'intent all'ora selezionata, altrimenti la notifica non verrà reinviata.

UTILITIES

Utilizzo il package Utilities per racchiudere diverse classi utilizzate in momenti diversi dell'applicazione.

La classe Utility contiene tutte le stringhe e le variabili statiche richiamate in diverse parti del workflow, inoltre contiene le funzioni che, data una variabile Date, restituiscono il primo e l'ultimo giorno di un determinato periodo, in più contiene anche la funzione che permette di generare in modo pseudo randomico i reports.

La classe Converters contiene le funzioni che permettono di fare il casting di un tipo di variabile da String a Date, o da Date a Long e viceversa, queste vengono utilizzate dal database per memorizzare le variabili di tipo Date e vengono usate anche in diverse parti del codice.

La classe OnSwipeTouchListener permette di osservare e gestire gli swipe verso destra e verso sinistra nei fragment Home e diario.

La classe SnackbarUndo viene utilizzata quando si elimina un report, infatti questo viene prima memorizzato nella classe stessa, in seguito la classe mostra il messaggio che riferisce che tale report è stato eliminato. Se si tocca il bottone "Cancella operazione" che compare sul messaggio, allora il report verrà ripristinato, altrimenti dopo qualche secondo verrà eliminato definitivamente.

SCELTE IMPLEMENTATIVE

Elenco qua alcune delle scelte implementative e accorgimenti che ho avuto durante lo sviluppo software.

- Durante la creazione dei grafici nel fragment Statistiche, inizialmente chiedevo al database le medie dei valori su base giornaliera, ottenevo così, però, dei grafici disordinati e pieni di dati. Solo in seguito, ho deciso di richiedere i dati su due scale diverse in base al periodo di visualizzazione, infatti se viene visualizzata la settimana o il mese le medie sono su base giornaliera, negli altri casi mensili.

- I grafici inerenti ai valori della pressione e della glicemia, raccogliendo due tipi di dato hanno bisogno di un metodo di rappresentazione consono, per questo ho usato un istogramma invece che un diagramma cartesiano, per permette la visualizzazione di due valori che vengono generalmente registrati nello stesso momento e poi analizzati in coppia.

- Ho deciso di utilizzare come layout dell'applicazione un design semplice e dei colori verdi, tipici delle applicazioni inerenti alla salute e al benessere.

- Dato che il numero delle righe nel database dei settings è fisso, avrei potuto utilizzare come primary key il campo settings_valore, ma ho preferito utilizzare un id intero che si auto incrementa, rendendo così l'entità più ordinata ed evitare il riutilizzo del campo del valore.

- Ho deciso di memorizzare l'orario per la gestione delle notifiche in formato intero (un intero per il valore delle ore e un intero per i minuti) e per memorizzare l'aggiunta dei report in formato stringa (hh:mm) inizialmente per comodità, poi però mi sono ricreduta e ho notato che implementare la gestione del tempo tramite un calendario avrebbe creato molti meno problemi.

REPORT E MONTE ORE

Per lo sviluppo software in tutte le sue componenti ho utilizzato circa 260 ore di lavoro ed ho riportato ogni modifica del progetto nella tabella sottostante.

Data	**Ore di lavoro**	**Operazioni svolte**	**Problemi riscontrati**	**Versione**
24/04/2020	2,00	Creazione cartelle e documenti Raccolta dati e informazioni riguardanti i dati da voler monitorare nella app Creazione abbozzi di grafica		
26/04/2020	5,00	Ricerca e ripasso sul corretto funzionamento di una activity Creazione di fragment e utilizzo di navigation Inizio scrittura del codice: Main Activity e bottom_navigation_menu	Problemi con l'inclusione di alcune librerie Errori stupidi dovuti alla mia ignoranza in materia Sbagliata creazione del menu	0
27/04/2020	2,00	Aggiunta dei fragment e delle classi per il diario e le statistiche Inizialmente sono riuscita a far funzionare il bottom_navigation menu, solo dopo ho deciso di cambiare le cose e utilizzare un navigation fragment	Non so per quale motivo l'app va in crush ora che aggiunto il navigation fragment, inizialmente con la bottom_navigation_menu riuscivo a switchare tra i 3 fragment nonostante questi fossero effettivamente sovrapposti tra loro Forse dovrei creare 3 activiy differenti e non usare dei fragment?	1
01/05/2020	1,00	Ho tolto la navigation Ho sistemato i 3 bottoni ed ora cambia il fragment correttamente senza che si sovrappongano e aggiunge correttamente in backstack	Ho provato a rinominare un pacchetto ma è esploso tutto quindi forse è il caso che io guardi come si fa	2
27/09/2020	2,00	Ho ridisegnato la app Ho capito che per prima cosa mi conviene capire come far funzionare le varie "parti" del progetto, quindi ho fatto una ricerca sui database room		TestDB_1
28/09/2020	3,00	Ho creato le istanze del database e ho provato a far funzionare i primi metodi di inserimento e visualizzazione dei dati	Non mi è chiaro come @Query permetta di restituire i dati del database Mi devo informare su	TestDB_2

			come posso visualizzare i report in una lista	
29/09/2020	4,00	Mi sono documentata su vari usi del database e le varie classi da implementare per il suo corretto funzionamento Ho testato illancio di una nuova activity e ho utilizzato la stessa metodologia (tramite intent) per il test dei DB	Quando provo a lanciare tramite un intent l'activity che permette di inserire i dati nel database va in crush	TestDB_3 TestDB_4 TestActivity_1
03/10/2020	2,00	Ho fatto ordine tra le cartelle di test e creato i vari brach su Github		
04/10/2020	4,00	Nella Main Activity viene visualizzata una lista di elementi fasulli, ho aggiunto il fragment per la visualizzazione della lista nella Home Ho aggiunto il fab che mi permette di lanciare una nuova activity che ha l'utilità di raccogliere i dati da inserire nel db	Il bottone per aggiungere un nuovo report funziona ma va in crash quando provo ad aggiungere al db i dati	TestDB 7 TestDB 8 TestDB 9
05/10/2020	3,00	Ho fatto l'accesso al db tramite AsyncTask e non nel main thread		TestDB 10
06/10/2020	2,00	Mi sono resa conto di aver perso gli aggiornamenti dell'ultimo update, ho cercato di riscrivere il codice, fallendo	Non riesco a ricreare la classe AsyncTask nel modo corretto	TestDB 10
09/10/2020	3,00	Ho risolto il problema precedente creado la classe ReportViewModel che gestisce le funzioni asincrone che permettono le modifiche del DB fuori dal Main Thread Ho aggiunto la classe Report List View che serve a gestire la visualizzazione della lista di report	Qualcosa non va, la lista non viene mostrata	TestDB 11
10/10/2020	3,00	Sono tornata sui miei passi e ho sistemato l'aggiunta del report, ora funziona correttamente (testato con sqllite)		PHM 1 PHM 2
11/10/2020	3,00	Ho aggiunto la ReportListAdapter e la ReportViewHolder che permettono la visualizzazione dei report nel XML della main activity (viene visualizzata solo la descrizione)		PHM 3
15/10/2020	3,00	Ora funziona l'update e il delete, ho anche aggiunto la snackbar che permette di eliminare l'ultima operazione di delete		PHM 4
18/10/2020	5,00	Ho modificato l'entità del report per fare in modo che conenga tutti i campi necessari per registrare il report Ho modificato new_report_activity.xml per fare in modo che ci siano i campi adeguati per inserire i valori ho fatto la funzione check2input che controlla che siano stati inseriri almeno due valori nel nuovo report Ho inserito il capo "data" che contiene la data e l'ora dell'inserimento del report (viene salvata la current data)	La recyclerview mostra la lista dei report ma mostra solo il capo "nota" Bisogna sistemare il file delle stringhe	PHM 5 PHM 6 PHM 7 TestFullReport TestFullReport1
19/10/2020	7,00	Ho sistemato la parte grafica in new_report_activity.xml e in list_report.xml Ho aggiornato le funzioni di update/delete/create di un report e i rispettivi controlli		PHM 8 PHM 9 PHM 10
20/10/2020	2,00	Ho aggiunto lo swipe left e right nella Main Activity		PHM 11

21/10/2020	6,00	Ho aggiunto il fragment TodayReport che mostra la data attuale	Non riesco a modificare la data quando faccio swipe left e right, devo prima capire come gestire DataFormat	PHM13
27/10/2020	5,00	Ho corretto la visualizzazione delle date nel TodayReport, ora anche swipe left e right funziona correttamente e viene visualizzata la lista dei report odierni nella main activity Vengono aggiornate bene le schermate di visualizzazione del TodayReport, ora la gestione e visualizzazione dei report è efficiente		PHM 14 PHM 15 PHM 16
28/10/2020	8,00	Ho aggiunto il menu Quello che prima era Activity Main ora è HomeFragment		PHM17
29/10/2020	8,00	Modifiche grafiche e aggiunta drawable Ho aggiunto il calendario nel fragment nel diario Aggiunta la lista dei report in ordine cronologico Aggiunta la visualizzazione dei report del giorno cliccato nel calendario	Bisogna sistemare il "back" della barra in alto	PHM18 PHM19
31/10/2020	2,00	Test di Material Calendar View	Non so, non va niente, aiutone	PHM20
03/11/2020	4,00	Ho sistemato la gestione del "tempo" nel database, ora salvo le date tramite un timestamp		PHM21 TestChart
04/11/2020	6,00	Ho creato il fragment delle statistiche, ho testato e inserito i grafici	Devo capire come ottenere il range della settimana, mese, anno	PHM22 TestPeriodo
05/11/2020	6,00	Ho aggiunto funzioni per ottenere il range della settimana/mese/anno Ora viene visualizzato correttamente il numero di report inseriti in quel lasso di tempo Viene visualizzato correttamente il grafico a torta in base ai report aggiunti, ho anche sistemato un bug nel salvataggio del giorno quando si modificava un report	Non mostra i report di una data specifica, solo di quella attuale Devo sistemare la visualizzazione del grafico e la distribuzione del codice tra il fragment e il model	PHM23 TestPeriodo PHM24 PHM25
06/11/2020	4,00	Ho testato il MaterialCalendarView e ho provato a integrarlo nel progratto Il calendario viene visualizzato correttamente Le icone degli eventi vengono visualizzati correttamente	Errore di visualizzazione del calendario Bisogna aggiungere gli eventi	PHM26 MaterialCalendarViewTest PHM27
07/11/2020	8,00	Ho creato le strutture per poter visualizzare i grafici dei singoli valori in base al periodo da visualizzare	Ci sono degli errori nella creazione della lista di tipo List<AVG> che ritorna le medie dei valori nel periodo indicato	PHM28 PHM29
08/11/2020	3,00	Ora la schermata delle statistiche viene visualizzata con i grafici corretti		PHM29
09/11/2020	5,00	Ho creato un generatore di report random per testare meglio la app Ho iniziato a sviluppare le impostazioni	Le date non sono molto troppo randomiche	PHM30 PHM31 PHM32
10/11/2020	6,00	Ho sistemato l'icona dell'ingranaggio nell'activity bar, le impostazioni si aprono correttamente Sto tentando di creare dei DatePicker e TimePicker per facilitare il settaggio dei dati nelle impostazioni	Non so se sviluppare con PreferencesCompact o no	PHM33 PHM34

11/11/2020	7,00	Ho creato parzialmente le impostazioni (la parte dei valori)		PHM35
12/11/2020	8,00	Ho finito di sistemare la schermata delle impostazioni, che ora è terminata e funziona e registra i valori correttamente nel database Nel database ho aggiunto i valori dei settings e delle notifiche		PHM36
13/11/2020	2,00	Ho iniziato a creare il filtro nel fragment diario	non vengono visualizzati I report dei singoli giorni con il filtro aggiornato	PHM37
15/11/2020	3,00	Ho sistemato le impostazioni riguardati la pressione nel database		PHM37
16/11/2020	7,00	Ho aggiornato la visualizzazione dei grafici con I nuovi dati della pressione aggiornati, ho messo un barchart invece che un linechart		PHM38
17/11/2020	2,00	Ho modificato alcune query in rawquery, in modo che queste vengano create dinamicamente in base al valore cercato	nella schermata delle statistiche, se passo dalla visualizzazione "settimana" a "tutti" va in crash	PHM39
18/11/2020	8,00	miglioramenti dei grafici (asse x, visualizzazione divisa per periodi specifici - ad esempio: giorno, settimana, mese)	Ho provato a chiedere al database di restituirmi il valore Count di determinati report ma da errore come se provassi ad accedere al db dal main thread	PHM40 PHM41
19/11/2020	6,00	Sono ritornata sui miei passi e ho fatto delle sistemazioni grafiche per poi nei giorni successivi sistemare anche il codice e risolvere il problema di ieri in un modo più efficace - ORA LA PARTE GRAFICA E' OKAY		PersonalHealthMonitor
20/11/2020	3,00	Ho sistemato la visualizzazione del fragment Home e ho trovato un modo per chiedere al database di restituirmi un valore primario Ho aggiunto i controlli all'input dei dati, ora posso aggiungere solo dati in un certo range numerico		PersonalHealthMonitor1
21/11/2020	5,00	Ho sistemato il fragment del diario, ora sto sistemando quello delle statistiche		PersonalHealthMonitor2
22/11/2020	6,00		Mi trovo nella situazione di dover richiedere un valore intero al database, normalmente quando faccio una richiesta al db mi faccio restituire un LiveData, in questo modo evito problemi del tipo "Cannot access database on the main thread since it may potentially lock the UI for a long period of time." Solo che questa volta, non posso utilizzare	PersonalHealthMonitor2

			un LiveData, perché le righe successive alla richiesta tentano di "elaborare" il dato richiesto, senza che questo sia effettivamente presente, tale per cui mi ritrovo in una situazione di errore del tipo "stai tentando di usare un valore null".	
23/11/2020	4,00	Ho cambiato nell'entità dei reports la modalità con la quale si salva la data di immissione del report, ho rimesso Date come inizialmente avevo fatto	Non riesco ancora a risolvere il problema descritto precedentemente	PersonalHealthMonitor3
24/11/2020	8,00	Ho risolto il problema precedente utilizzando una classe AsyncTask, sto programmando il funzionamento di questa per permettere la corretta visualizzazione dei grafici		PersonalHealthMonitor4 PersonalHealthMonitor5
25/11/2020	5,00	Ho sistemato la visualizzazione dei grafici di tipo LineChart, ovvero i grafici del battito cardiaco e della temperatura Ho sistemato la visualizzazione di tutti i grafici	Devo sistemare anche i grafici di tipo Bar Chart	PersonalHealthMonitor6
26/11/2020	6,00	Ho sistemato tutti i vari fragment	Devo implementare il bottone dei filtri	PersonalHealthMonitor7
27/11/2020	4,00	Ho implementato le funzioni che gestiscono il filtro e gli annessi bottoni e finestre di dialogo Ho fatto una ricerca/iniziato a sviluppare le notifiche	Devo implementare e testare le notifiche	PersonalHealthMonitor8 PersonalHealthMonitor9 PersonalHealthMonitor10 PersonalHealthMonitor11
29/11/2020	1,00	Ho creato un database a parte per la gestione delle notifiche Ho creato la classe notification che estende il broadcast receiver per poter inviare le notifiche giornalmente	Non vengono visualizzate le notifiche	PersonalHealthMonitor12
30/11/2020	3,00	Ho iniziato a testare le notifiche (giornaliere)		TestNotification
01/12/2020	4,00	Ora le notifiche giornaliere vengono visualizzate ogni minuto (per testarle)	Devo implementare un modo per visualizzare le notifiche sui valori	PersonalHealthMonitor13
02/12/2020	7,00	Ho provato a far visualizzare correttamente le notifiche giornaliere	Non riesco ad accedere alle informazioni del DB mentre l'applicazione non è ON	PersonalHealthMonitor14

03/12/2020	6,00	Ho lavorato per sistemare la notifica del report giornaliero	Devo fare apparire la notifica in caso di dati warning	PersonalHealthMonitor15 PersonalHealthMonitor16
04/12/2020	6,00	Ora vengono visualizzate correttamente le notifiche	Bisogna sistemare il pulsante "Rimanda" nella notifica giornaliera	PersonalHealthMonitor17
05/12/2020	5,00	Correzione visualizzazione notifiche e sviluppo del pulsante "Rimanda" Sistemato i colori e parti grafiche	Correzioni grafiche	PersonalHealthMonitor18 PersonalHealthMonitor19 PersonalHealthMonitor20
06/12/2020	5,00	Ho creato i layout landscape		Personal Health Monitor
07/12/2020	3,00	Ho sistemato la documentazione		Personal Health Monitor
08/12/2020	5,00	Corretto un bug nelle notifiche Sistemato la documentazione		Personal Health Monitor
09/12/2020	5,00	Sistemato la documentazione		Personal Health Monitor
Totale ore	261			

SOFTWARE E METODI DI CONDIVISIONE CODICE

ANDROID STUDIO

Android Studio è un ambiente di sviluppo integrato (IDE) per lo sviluppo per la piattaforma Android. Basato sul software di JetBrains IntelliJ IDEA, Android Studio è stato progettato specificamente per lo sviluppo di applicazioni Android. È disponibile il download su Windows, Mac OS X e Linux, e sostituisce gli Android Development Tools (ADT) di Eclipse, diventando l'IDE primario di Google per lo sviluppo nativo di applicazioni Android. Ho utilizzato questo software per lo sviluppo e creazione dell'applicazione.

DROPBOX

Dropbox è un servizio di file hosting gestito dalla società californiana Dropbox Inc., che offre cloud storage, sincronizzazione automatica dei file, cloud personale e software client. Ho utilizzato questo software per la sincronizzazione di file e cartelle via cloud per avere una raccolta dati sempre disponibile, anche in più dispositivi.

GITHUB

GitHub è un servizio di hosting per progetti software. Il nome deriva dal fatto che "GitHub" è una implementazione dello strumento di controllo versione distribuito Git. Ho utilizzato questo software per tenere sempre aggiornati i file del codice del progetto, ed eventualmente creare più fasi/versioni dello stesso.

DB BROWSER PER SQLITE

DB Browser per SQLite (DB4S) è uno strumento di alta qualità, visivo e open source per creare, progettare e modificare file di database compatibili con SQLite.

FONTI

- Stack Overflow: è un sito web che fa parte della rete Stack Exchange in cui si possono porre domande riguardo a vasti argomenti di programmazione.
- My personal Trainer: è una pagina web che fornisce informazioni di tipo sanitario. L'ho usata per fare ircerca riguardante i valori da inserire nei report
- Slide del corso di Laboratorio di Applicazioni Mobili