Progetto del Corso di Programmazione ad Oggetti
a.a. 2016 – 2017

Life Fit

Relazione di Riccardo Giorato
Matricola 1122158
Università di Padova

Contenuti relazione

1) Abstract

Il progetto si propone di realizzare un'applicazione in grado di gestire le iscrizioni di Utenti di una famiglia e le relative attività di movimento e di sonno registrate da activity tracker come Fitbit o MiFit.
Gli utenti vengono inseriti dall'amministratore; i dati possono essere inseriti dall'amministratore o dagli utenti adulti. Tutti gli utenti (i diversi membri della famiglia) potranno interrogare i propri dati (ad esempio la classifica settimanale dei passi effettuati) e confrontarli con quelli degli altri utenti; potranno inoltre visualizzare (tramite un calendario) delle statistiche giornaliere su quantità passi e ore di sonno.
E' possibile personalizzare il livello di informazioni disponibili per "tipologia" di utente ed è sempre possibile aggiornare i propri dati personali.

2) Parte Logica

Per evitare di generare confusione tra parte logica e parte grafica il progetto è stato sviluppato non tenendo conto dell'eventuale interfaccia grafica successivamente implementata e rispettando il più possibile i principi della programmazione ad oggetti dividendo parte quindi la Gui dal Modello mentre il controller è integrato con la parte di interfaccia Qt.
Il progetto presenta complessivamente due gerarchie nella parte logica che rappresentano i vari tipi di attività e gli utenti.

3) Gerarchia di Utente

La base della gerarchia è la classe Utente, definita astratta in quanto il distruttore è virtuale puro e possedendo funzioni polimorfe pure descritte nella sezione 6.

- Bambino: può solamente interrogare le informazioni relative all'ultimo giorno inserito dai genitori, non può aggiornare i suoi dati personale né aggiungere ulteriori dati
- Adolescente: può vedere tutte le informazioni relative alla sua attività (tramite calendario) e può visualizzare il suo storico completo
- Adulto: può visualizzare tutte le informazioni inserite ed inoltre può caricare dall'interfaccia nuovi dati senza doverlo richiedere all'amministratore

4) Gerarchia di Attività

La classe base è att_base con distruttore virtuale e metodo virtuale puro calorie() che su alcune attività come il sonno deve essere calcolato in modo differente.
Le classi derivate da att_base sono:

- att_sonno contenente informazioni relative alla qualità del sonno e tempo trascorso a letto e a dormire
- att_mov contenente numero di passi, piani, distanza e minuti attivi e non attivi totali.

Sarà possibile quindi aggiungere nuove attività in futuro oppure implementare visualizzazioni di informazioni sulle attività.

5) Contenitore Attività

Le attività sono memorizzate dentro gli oggetti giorno, ogni giorno possiede una data QDate e due oggetti, uno per att_sonno e uno per att_mov.
I giorni sono inseriti in una mappa formata dalla coppia QDate e giorno nella classe utente ossia base gerarchia di utenti.

6) Descrizione dell'uso di codice polimorfo

Nella classe utente, per non creare memory leak è stato definito il distruttore virtuale, in modo che quando si distrugge un puntatore a utente viene richiamato il distruttore corretto.
Oltre a questo l'utente possiede la funzione polimorfa clone per creare una copia dell'oggetto restituendo un puntatore di questa copia utilizzato nel file ui_user.cpp nella funzione on_saveUser_clicked().

Sono inoltre realizzate altre funzioni nella gerarchia sempre virtuali pure su utente come bool aggiungiAttivita(), int passiConsigliati(), bool settingsEnabled(), Qstring nomeGruppo(), int codiceGruppo(), Qcolor coloreGruppo() e infine bool viewCalendar.

Queste funzioni ritornano informazioni riguardo le differenti aspettative di attività fisica e sonno in base alla categoria dell'utente (esempio. un bambino deve fare meno passi di un adulto) oltre a definire permessi dell'utente che sono utilizzati nella parte grafica per abilitare o disabilitare funzionalità ad esempio in ui_user.cpp nella funzione loadUserOnUi.

7) GUI

L'interfaccia è rappresentata da tre finestre. La prima che si incontra chiamata dal main.cpp è il LoginForm per permettere login utenti o dell'amministratore.
Quindi si passa a Ui_Admin per aggiungere utenti, eliminarli o modificarli oppure Ui_User dove l'utente visualizza le proprie informazioni con le sue autorizzazioni possibili.

Da Ui_Admin si può passare a Ui_User con visualizzazione assoluta admin oppure alla Register per la registrazione di un nuovo utente nel sistema.

Infine da Ui_User si può effettuare il logout e tornare a LoginForm.

8) Istruzioni per l'utilizzo

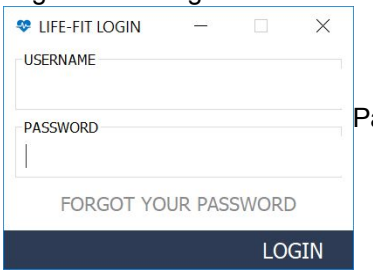
Il programma risulta funzionante compile time e run time attraverso l'uso dei comandi:

1. qmake progetto.pro
2. make
3. ./progetto_p2

Viene fornito l'utente **Amministratore** col quale si possono aggiungere nuovi utenti:

- **Username:"root"**
- **Password:"root"**

Sono stati consegnati anche file xml come input per i differenti utenti con informazioni diverse per attività negli ultimi 120 giorni con nome att1.xml, att2.xml e att3.xml nella cartella xml_input.



Pannello per il login utenti o amministratori a sinistra.

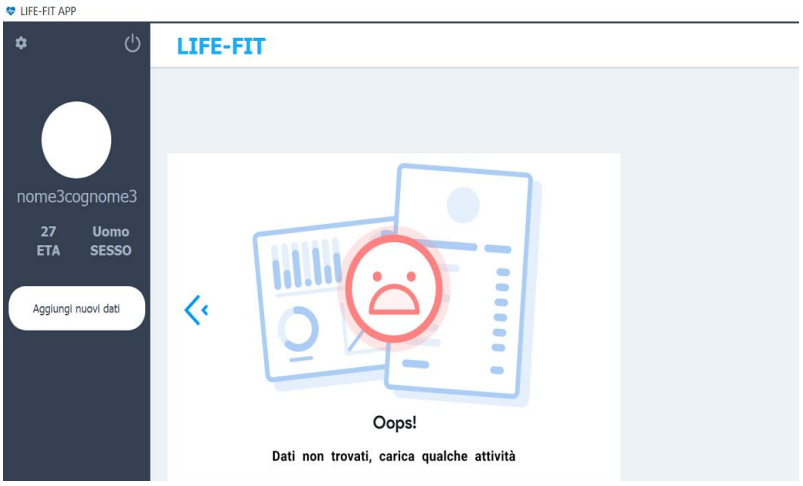
Pannello amministratore, dopo aver aggiunto vari utenti è possibile ai successivi login fare doppio click sull'utente nella lista generata per effettuare visualizzazione e modifiche dell' utente in Modalità Admin View.

Per eliminare un utente selezionarlo dalla lista e premere pulsante rosso.

Ui_User Modalità Admin View a destra. Aggiunta dei dati sempre possibile con questa modalità con qualsiasi tipo d'utente.

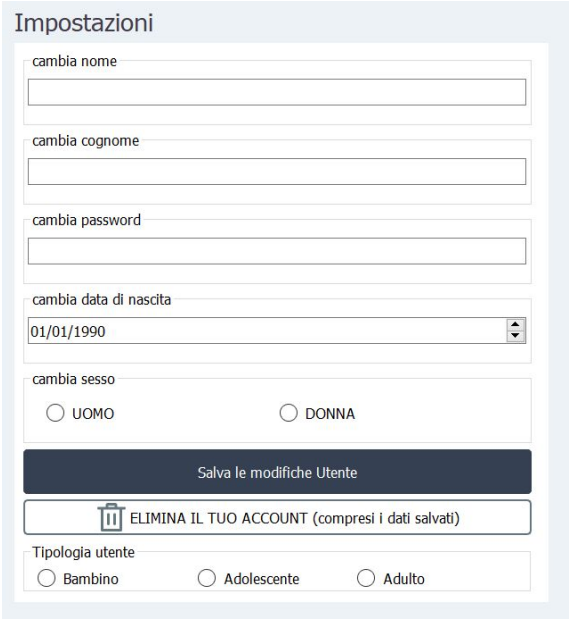
Ui_User modalità utente a destra.

Cliccando sul pulsante impostazioni in alto a sinistra si accede alle impostazioni dell'utente.
Cliccando sul pulsante in alto a sinistra si effettua il logut.



Impostazioni utente a destra, visto in modalità admin è quindi possibile cambiare pure il tipo dell'utente tra quelli implementati attualmente.

In modalità utente normale non è possibile modificare la propria tipologia, questo groupbox è quindi nascosto.



9) Ore utilizzate:

- progettazione: 5h
- progettazione grafica (inclusa la scelta dei layout): 5h
- scrittura codice: 14h
- scrittura gui: 36h
- test: 3h
- debug(compresa analisi Clang Static): 1h

Totale: 64h

10) Ambiente di sviluppo

Versione QtCreator: 4.3.0
Versione Qt: 5.9.1
Versione GCC : 4.9.2
Qmake: 3.0
O.S.: Windows 10 PRO