Relazione progetto di Programmazione ad Oggetti (2014-2015)

LinQedIn

Miki Violetto
1029140

OS: Ubuntu 12.04 LTS - gcc 4.6.3
Qt Creator 3.2.1 (open source) – basato su Qt 5.3.2 (GCC 4.6.1, 64 bit)

Credenziali d'accesso

Basic : miki@libero.it
Business : francesco.rossi@aruba.com
Executive : funny_fox@gmail.com

Sommario

LinkedIn è il principale servizio web di rete sociale per contatti professionali, gratuito ma con servizi opzionali a pagamento. Lo scopo del progetto è lo sviluppo in C++/Qt di un sistema minimale per l'amministrazione ed utilizzo tramite interfaccia utente grafica di un (piccolo) database di contatti professionali ispirato a LinkedIn.

Requisiti funzionali per l'utente

Un utente è caratterizzato almeno da:

1. Profilo: contetente le informazioni personali (ad esempio: nome, cognome, residenza, ecc.).
2. Tipologia dell'account di iscrizione: Basic, Business o Executive.
3. Rete di contatti professionali con altri utenti.

La classe **InfoPersonali** contiene tutte le informazioni personali dell'utente:
email, nome, cognome, residenza, sesso, data di nascita, luogo di nascita, lavoro, titolo di studio e numero telefonico.
L'email e il numero telefonico vengono registrati con opportune classi che ne controllano la correttezza sintattica.

Gerarchia di utenti

Le diverse tipologie di account determinano diversi tipi di utenti che devono essere modellati mediante una opportuna gerarchia di classi.
Si è deciso di creare una gerarchia verticale dove ogni tipo di utente è rappresentato da una classe derivata da quella di un utente che ha meno permessi:

1. L' **UtenteBasic** permette solo la ricerca di nome, cognome ed email.
2. L'**UtenteBusiness** può in aggiunta effettuare la ricerca di residenza, sesso, data e luogo di nascita.
3. L'**UtenteExecutive** può in aggiunta effettuare la ricerca del numero telefonico, lavoro e titolo di studio, oltre a permettere una ricerca sui contatti degli utenti

La classe **Utente** contiente le infoPersonali, la reteContatti, un booleano per capire se le informazioni sono state nascoste.
Viene implementato il metodo *QString getTipo() const* che ritorna tramite una QString il tipo dell'utente il quale è confrontabile con la variabile QString statica presente in ogni classe derivata di utente.

Database

Il database è creato utilizzando la classe QMap. Un contenitore della QT la cui implementazione e` quella di un albero binario di ricerca in cui viene ogni nodo punta ad una chiave univoca.
Come chiave si utilizza l'email, dopo aver ridefinito l'operator<;
Si utilizza la classe **PointerUtente** per offrire un puntatore intelligente all'utente: ogni volta che ci si riferisce ad un utente si utilizza questo puntatore per ovviare al problema del cambio del puntatore di utente (dato dal cambio del tipo di utente). Un secondo motivo del suo utilizzo lo si nota quando viene utilizzato per mantenere controllato il numero di utenti da e a quell'utente e, nel caso dell'eliminazione di un utente permette un'eliminazione che mantiene la consistenza del database.
La particolarità dell'implementazione del contatto permette una relazione a verso unico, cioè l'utente A può avere come contatto l'utente B senza che B abbia A come contatto.

Ricerca

Ogni tipo di utente implementa la ricerca con una classe apposita che contiene i tipi di ricerca che può eseguire. La ricerca accetta una ricerca precedente, e permette di concatenare ricerche.

Admin

La classe **Admin** permette le seguenti funzionalità:

- inserimento di un nuovo utente
- rimozione di un utente
- cambio di tipologia di account (Basic, Business, Executive) di un utente
- caricamento e salvataggio del database in un apposito file xml

Client

La classe **Client** permette le seguenti funzionalità`:

- aggiornamento del proprio profilo
- inserimento di un contatto nella propria rete
- rimozione di un contatto dalla propria rete
- ricerca nel database di un utente (diversa per ogni tipologia di account)

Interfaccia grafica

La componente grafica è composta da principalmente da quattro parti: due bottoni che permettono di scegliere dove accedere e due aree sottostanti.
Le aree sottostanti sono suddivise verticalmente in:

- Su admin la parte destra presenta la lista delle operazioni eseguibili, mentre la parte destra le visualizza; questo risulta in un' astrazione che stona, in quanto molte operazioni di admin di solito sono nascoste all' utente.
- Su client si è cercato di riportare piu' fedelmente l' aspetto social a cui siamo abituati, l'area a sinistra contiene un link al profilo e la lista dei propri contatti, mentre quella a destra contiene la barra di ricerca e la scheda che si sta guardando.

Prestazioni

E` stata curata anche l'ottimizzazione del programma, sebbene non fosse una priorità, tramite alcuni accorgimenti.
In Utente si è implementato un costruttore protetto *Utente(InfoPersonali &)* per velocizzare il caricamento del database.
L'utilizzo di PointerUtente come classe per mantenere la consistenza del database ha portato ad un aumento delle operazioni da eseguire nell'inserimento e rimozione di un contatto, come nella rimozione di un utente, ma si è preferito la modifica istantanea di questi legami alla velocita' di esecuzione.
Riguardo la gestione della memoria, soprattutto della parte Qt, si è prestata attenzione alla distruzione degli oggetti creati: la Qt implementa nel distruttore una distruzione profonda di tutti i figli, quindi si è prestata attenzione al layout, la cui distruzione non viene eseguita sui figli, e sugli oggetti senza padre o che devono venire eliminati senza eliminare il padre.

Ambiente di sviluppo

Il progetto è stato realizzato sull'ambiente Linux
Il compilatore utilizzato è il GCC 4.6.1 e utilizza le librerie di Qt 5.3.2 (comando qmake-qt532 nelle macchine del laboratorio).

Compilazione dei sorgenti

Visto l'utilizzo della classe QDebug nel progetto, viene fornito un makefile con la riga aggiuntiva

```
DEFINES += -DQT_NO_DEBUG_OUTPUT
```

e il file .pro, quindi sara` necessario solo esequire il comando make per compilarlo.
Il file database.xml viene fornito un utente per tipo gia` presente, e viene cercato nella stessa posizione.