Progetto PROGRAMMAZIONE ad OGGETTI

anno 2014 - 2015

Davide Tommasin - Matricola 1073541

INTRODUZIONE

Lo scopo di questo progetto è sviluppare in linguaggio C++ un sistema basato su LinkedIn, per l'amministrazione e l'utilizzo di un database tramite un interfaccia grafica.

- Ci sono tre tipologie di account: Basic, Business e Executive

- Ci sono due tipi differenti di accesso: Utente e Amministratore

Il sistema gestisce le informazioni relative ai vari utenti iscritti, e memorizza tali informazioni in un database, dove i vari utenti e l'amministratore andranno a visualizzare o gestire i vari iscritti.

Per sviluppare ciò è stato utilizzato:

- IDE **QtCreator 3.5.0** Based on **Qt 5.5.0** (**Clang 6.0** (Apple), 64 bit)

- Sistema Operativo: **OS X** Yosemite Versione 10.10.5

PARTE LOGICA

Per sviluppare la parte *logica* del sistema sono state utilizzate le seguenti classi:

- Utente
- UtenteBasic
- UtenteBusiness
- UtenteExecutive
- Profilo
- Client
- Admin
- DataBase

Classe "*Utente*"

Nella classe *Utente* è principalmente memorizzato l'username (di tipo QString), il profilo dell'utente(di tipo *Profilo*) e i contatti della rete dell'utente (di tipo QStringList).
La classe *Utente* prevede un solo costruttore a un parametro in quanto l'username deve sempre essere presente essendo il nome univoco e non nullo presente nel database.
La classe *Utente* ha un campo di tipo *Profilo* dove sono memorizzati tutti i dati di un utente. Per il campo di tipo profilo vengono messi a disposizione tutti i metodi set e get per visualizzare o modificare i vari dati.
Nella classe *Utente* è presente un campo contatti, è la rete contatti dell'utente.
Per ultimo la classe *Utente* prevede un metodo virtuale puro stampaInfo(bool&,bool&) il quale verrà implementato dalle classi derivate da *Utente.* Con questo metodo virtuale puro sarà impossibile creare oggetti di tipo *Utente.*

Classe "*Utente Basic*"

La classe *UtenteBasic* deriva la classe *Utente*, ereditando i suoi campi e metodi e implementando il metodo stampaInfo(bool&,bool&), così sarà possibile creare oggetti di tipo *UtenteBasic.*
UtenteBasic implementa il metodo stampaInfo(bool&,bool&) in modo da dare dei privilegi diversi dalle altre due classi *UtenteBusiness* e *UtenteExecutive.*

Classe "Utente Business"

La classe *UtenteBusiness* deriva la classe *Utente*, anch'essa come la classe *UtenteBasic* eredita i campi e metodi e implementa il metodo stampaInfo(bool&,bool&), così sarà possibile creare oggetti di tipo *UtenteBusiness.*
UtenteBusiness implementa il metodo stampaInfo(bool&,bool&) in modo da dare dei privilegi diversi dalle altre due classi *UtenteBasic* e *UtenteExecutive.*

Classe “Utente Executive”

La classe *UtenteExecutive* deriva la classe *Utente*, anch'essa come la classe *UtenteBasic* e *UtenteBusiness* eredita i campi e metodi e implementa il metodo stampaInfo(bool&,bool&), così sarà possibile creare oggetti di tipo *UtenteExecutive.*
UtenteExecutive implementa il metodo stampaInfo(bool&,bool&) in modo da dare dei privilegi diversi dalle altre due classi *UtenteBusic* e *UtenteBusiness*

Classe “Profilo”

Nella casse *Profilo* sono presenti le informazioni relative ad un utente (nome, cognome, data di nascita, residenza, professione, titolo di studio, le proprie esperienze, le proprie competenze e le lingue conosciute). Tutti questi campi sono pubblici in quanto la classe *Profilo* verra utilizzata solamente come campo privato della classe *Utente.*
È presente solo il costruttore di default in quanto tutti i campi della classe *Profilo* sono modificabili con i metodi set della classe *Utente*.

Classe “Client”

Un oggetto di classe *Client* è un’istanza di un utente di LinQedIn. La classe prevede un campo di tipo *DataBase*,* due campi di tipo *Utente**: utenteLoggato, che tiene traccia dell’utente che ha effettuato l’accesso in LinQedIn; utenteSelezionato è l’utente, che è stato cercato dall’utenteLoggato, nella rete di LinQedIn; e per ultimi sono presenti due campi stampaProfilo (di tipo bool) e stampaProfiloCompleto (di tipo bool), i quali sono settati dal metodo stampaInfoCercato() che effettuerà una chiamata polimorfa al metodo stampaInfo(bool&,bool&) dando così diversi privilegi all’utenteLoggato in base alla sua tipologia di account. Sono inoltre disponibili i vari metodi di modifica del profilo dell’utenteLoggato, e altri metodi di ricerca per cercare vari utenti nella rete di LinQedIn ed eventualmente aggiungerli o rimuoverli alla propria rete.

Classe “Admin”

Un oggetto di classe *Admin* rappresenta l’amministratore del database di LinQedIn, difatti ha come unico campo privato un puntatore al database, il quale viene inizializzato all’avvio dell’applicazione.
La funzione della classe *Admin* è appunto quella di gestire i vari utenti iscritti a LinQedIn, può gestire un solo utente alla volta (memorizzato nel campo *Utente** selezionato).
Questa classe comprende dei metodi per aggiungere, rimuovere o visualizzare i vari utenti, ed eventualmente cambiargli la tipologia di account.
Sono presenti anche due metodi di ricerca (uno per username, l’altro per nome e cognome) per cercare gli utenti presenti nel database.

Classe “DataBase”

Un oggetto di tipo *DataBase* rappresenta il database di LinQuedIn, dove verranno memorizzati i vari utenti iscritti tramite dei puntatori.
Nella classe *DataBase* sono presenti i metodi per caricare il database, per salvare le varie modifiche apportate al database e altri metodo di ricerca e aggiunta o rimozione dei vari utenti.
In questa classe è presente anche un distruttore il quale dealloca tutti gli utenti allocati sullo heap.

PARTE GRAFICA

Classi:

- LinqedinWidget
- AdminWidget
- LoginWidget
- RegistrazioneWidget
- ClientWidget
- ProfiloWidget
- ProfiloReteWidget

Classe "LinQedIn Widget"

La classe *LinqedinWidget* genera l'interfaccia iniziale del programma. Viene costruito un oggetto di tipo *DataBase* dove verrà utilizzato il metodo carica(), così facendo genererà il database di LinQedIn.
Da questa schermata iniziale si può selezionare con quale modalità di accesso (Utente o Amministratore) accedere a LinQuedin.

Classe "Admin Widget"

La classe *AdminWidget* crea l'interfaccia dell'amministratore di LinQedIn con la quale l'amministratore può andare ad apportare modifiche sul database cercando, modificando, inserendo o eliminando i vari utenti.

Classe "Login Widget"

La classe *LoginWidget* fornisce l'interfaccia grafica di login, dove si può accedere al programma inserendo il proprio username oppure si può essere ri indirizzati ad un form di registrazione.

Classe "RegistrazioneWidget"

La classe *RegistrazioneWidget* permette tramite dei form di iscriversi a LinQedIn.

Classe "Client Widget"

La classe *ClientWidget* crea l'interfaccia dell'utente di LinQedIn, userà il database creato in precedenza e come utente userà l'username inserito nel form di login precedente.
La classe consente di ricercare e aggiungere altri utenti presenti in LinQedIn e consente di modificare il proprio profilo per tenerlo aggiornato.

Classe "Profilo Widget"

La classe *ProfiloWidget* fa visualizzare le varie informazioni relative all'utente, ma non verrà visualizzata la rete dei contatti relativa all'utente.

Clase "Profilo Rete Widget"

La classe *ProfiloReteWidget* visualizza le varie informazioni relative all'utente, come nella classe *ProfiloWidget,* ma in questa classe viene visualizzata anche la rete dei contatti relativa all'utente.