Relazione LinQedIn Davide Rigoni

Versione Qt: 5.2.1

Ambiente di sviluppo: Qr Creator 3.0.1

Il progetto

Il progetto e' stato costruito seguendo il pattern MVC, separando la model dalla view con l'ausilio di una classe di controllo.

LinQedIn si puo' suddividere principalmente in 3 parti.

1. Login: Questa parte dell'applicazione permette ad un utente di loggarsi in LinQedIn tramite una apposita schermata. Il login pu' essere eseguito come Client, tramite email e password, o come Amministratore . Questa e' la prima finestra che appare quando il programma viene eseguito.
2. Parte amministrativa: Questa parte dell'applicazione permette:
    - Aggiungere o rimuovere utenti.
    - Modificare il tipo di iscrizione a LinQedIn di un utente.
    - Effettuare delle operazioni sul database.
    - Cercare degli utenti tramite Nome,Cognome ed Email nel database.
3. Parte Client: Questa parte permette:
    - Vedere e modificare le informazioni del proprio profilo .
    - Vedere e modificare la propria rete delle amicizie.
    - Cercare utenti in LinQedIn tramite delle opportune form.
    - Vedere i profili di utenti amici o cercati in LinQedIn.

Il progetto e' stato costruito cercando di modularizzare il pi' possibile le parti e di rendere il codice il piu' estensibile possibile.

Ogni classe creata ha dei metodi virtuali e dei costruttori che accettano puntatori ad oggetti in modo da poter usare il polimorfismo.

Ogni classe grafica possiede i campi dati puntatori agli oggetti grafici nella sua parte protetta in modo da permettere alle classi derivate di accere a questi elementi se necessario.

Gli utenti

Ogni utente deriva direttamente o indirettamente dalla classe base astratta AUser, la quale rappresenta la struttura di un utente con le sue funzioni minime.

Le classi derivate sono:

- FreeUser che deriva direttamente da AUser
- BusinessUser che deriva direttamente da AUser e si differenzia da FreeUser in

quanto e' un utente a pagamento con una funzione di ricerca piu' avanzata

- ExecutiveUser che deriva direttamente da BusinessUser in quanto anch'essa e' a pagamento e presenta una funzione di ricerca piu' avanzata di BusinessUser

La classe AUser presenta principlamente:

- Un campo dati pubblico puntatore ad un oggetto che rappresenta la lista degli amici
- Un campo dati pubblico puntatore ad un oggetto profilo
- Un campo dato privato che rappresenta la passwordCryptata
- Un metodo virtuale puro pubblico che implmenta la ricerca fatta dagli utenti
- Un costruttore che accetta come passaggio di parametri dei puntatori polimorfi all'oggetto profilo e all'oggetto che rappresenta la rete delle amicizie

Le classe derivate presentano principalmente una funzione di ricerca piu' evoluta.

Il Database

Il database e' stato costruito con l'idea che la sua funzione sia principalmente quella di gestire i problemi di salvataggio, caricamento e accesso ai dati.Per questo motivo non implementa nessun metodo di ricerca che non sia tramite la chiave della QMap rappresentata dalla email degli utenti.

E' stato scelto di adoperare un contenitore QMap in quanto permette di associare in modo univoco un puntatore polimorfo ad un utente a una chiave.

Parte LinQedIn Admin

Questa parte e' stata costruita utilizzando una parte model LinQedInAdmin, una parte di controllo LinQedInAndminController e una parte LinQedInAdminView.Tutte queste classi possiedono metodi virtuali.

La classe LinQedInAdmin possiede un costruttore ad un parametro che accetta un puntatore polimorfo ad un oggetto database e mette a disposizione tutte le operazioni che l'amministratore puo' compiere.

La classe LinQedInAndminController possiede un costruttore rispettivamente ad uno e a due parametri.Il primo accetta un puntatore polimorfo ad un oggetto LinQedInAdmin e il secondo che ha come parametro anche un puntatore ad un oggetto LinQedInAdminView.Questo per permettere una maggiore estendibilita' del codice.Il compito dell' LinQedInAndminController e' quello rilevare le signals dell'oggetto view, elaborare le informazioni tramite i metodi presenti nel LinQedInAdmin e aggiornare la view tramite i metodi messi a disposizione della classe LinQedInAdminView.

La classe LinQedInAdminView rappresenta la parte grafica della schermata dell'amministratore e dispone di metodi virtuali permettendo cosi' di essere estesa o modificata in futuro.

Parte LinQedIn Client

Questa parte e' stata costruita utilizzando una parte model LinQedInClient, una parte di controllo LinQedInClientController e una parte LinQedInClientView.Tutte queste classi possiedono metodi

virtuali.

La classe LinQedInClient possiede un costruttore a due parametri che accetta puntatori polimorfi rispettivamente ad un oggetto database ed ad un oggetto AUser.

La classe LinQedInClientController possiede un costruttore rispettivamente ad uno e a due parametri.Il primo accetta un puntatore polimorfo ad un oggetto LinQedInClient e il secondo che ha come parametro anche un puntatore ad un oggetto LinQedInClientView.Questo per permettere una maggiore estendibilita' del codice.Il compito dell' LinQedInClientController e' quello rilevare le signals dell'oggetto view, elaborare le informazioni tramite i metodi presenti nel LinQedInClient e aggiornare la view tramite i metodi messi a disposizione della classe LinQedInClientView.

La classe LinQedInClientView rappresenta la parte grafica della schermata dell'utente e dispone di metodi virtuali permettendo cosi' di essere estesa o modificata in futuro.Questa classe grafica e' formata da un oggetto QTabWidget che crea delle Tab che contengono altri oggetti view.Questo per permettere una magiore modularizzazione della parte grafica e una maggiore estensibilita' del codice.La classe LinQedInClientView dispone di piu' costruttori che permettono il passaggio di puntatori ad oggetti grafici UserContactsNetView,UserBaseInfView e UserSearchView.

La classe UserBaseInfView e' una classe che utilizza signal e slot per visualizzare il profilo dell'utente loggato e in caso anche per modificarlo. Questa classe e' utilizzata anche per visualizzare i profili degli utenti di linkedin.

La classe UserContactsNetView e' una classe che utilizza signal e slot per visualizzare la rete delle amicizie dell'utente loggato e in caso anche per modificarla.

La classe UserSearchView e' una classe che utilizza signal e slot per visualizzare la form di ricerca per ogni tipologia di utente FreeUser,BusinessUser,ExecutiveUser.

Parte LinQedIn Login

Questa parte e' stata costruita utilizzando una parte model LinQedInLogin, una parte di controllo LinQedInLoginController e una parte LinQedInLoginView.Tutte queste classi possiedono metodi virtuali.

La classe LinQedInLogin carica il database db.xml ed esegue il login dell'utente o dell'amministratore.

La classe LinQedInLoginController possiede un costruttore rispettivamente ad uno e a due parametri.Il primo accetta un puntatore polimorfo ad un oggetto LinQedInLogin e il secondo che ha come parametro anche un puntatore ad un oggetto LinQedInLoginView.Il compito dell' LinQedInClientController e' quello rilevare le signals dell'oggetto view, elaborare le informazioni tramite i metodi presenti nel LinQedInClient e aggiornare la view tramite i metodi messi a disposizione della classe LinQedInClientView.

La classe LinQedInClientView rappresenta la parte grafica della schermata del login e dispone di metodi virtuali permettendo cosi' di essere estesa o modificata in futuro.I puntatori agli oggetti grafici sono protetti in modo di poter rendere possibile l'accesso a quest'ultimi anche dalle classi derivate.

Altro

Assieme ai file e' stato consegnato anche un file db.xml che contiene il database. La password per tutti gli account e': d