Relazione sul progetto di Programmazione ad Oggetti a.a. 2014/2015

LinQuedIn

Simone Magagna
matricola: 1009467

Ambiente di sviluppo

- Compilatore: GCC 4.9.1, 64 bit
- Versione Qt Creator: 3.1.1
- Versione Qt: 5.3.0
- Sistema Operativo: Ubuntu 14.10

Materiale consegnato

La cartella consegnata nominata: “Simone Magagna progetto” contiene i seguenti elementi:

- LinQuedIn: cartella nella quale sono presenti tutti i file .h, .cpp e .ui e un file .pro necessario per la giusta compilazione del progetto
- database.xml: file contenente un database di prova nel linguaggio xml
- relazione.pdf: relazione sul progetto

Compilazione ed esecuzione

Attraverso la shell del Terminale posizionarsi all'interno della cartella “LinQuedIn” ed eseguire il comando qmake. Una volta generato il Makefile, sempre all'interno della cartella “LinQuedIn”, avviare il comando make. Questa operazione genererà un file eseguibile LinQuedIn che potrà essere avviato tramite un doppio click.

1. Introduzione

Il progetto in esame si propone modellare le principali funzioni offerte dalla piattaforma LinkedIn, servizio web di rete sociale per contatti professionali, gratuito ma con servizi a pagamento. Questi ultimi sono stati implementati mediate tre diverse tipologie d'utenza: Basic, Business ed Executive. A differenza della classe Basic, l'appartenenza alla classe Business ed Executive è a pagamento ma offre opzioni di ricerca di altri clienti iscritti maggiori.
Sono state rese disponibili le principali funzionalità sia del lato amministratore che del lato cliente. Un amministratore potrà inserire un nuovo cliente, rimuoverlo, visualizzare le sue principali generalità e modificargli in un secondo momento la tipologia d'utenza. Un cliente d'altro canto potrà visualizzare il proprio profilo, modificarlo e aggiornalo, aggiungere e rimuovere un cliente dalla propria rete di contatti e infine ricercare un utente all'interno del database e conoscerne il profilo in base alla propria tipologia d'utenza. Se l'utente che effettua la ricerca è Basic, avrà accesso solo all'username, nome, cognome e e-mail dell'utente cercato. Se è Business, potrà vedere questi campi dati e in più le esperienze professionali, le competenze lavorative, le lingue conosciute ed i titoli di studio. Infine se un utente appartiene alla categoria Executive, potrà visualizzare anche la sua rete di contatti.
Nella realizzazione del progetto si è cercato il più possibile di rispettare il pattern MWC: Model Wiew Controller. Esso consiste nel mantenere divisa la parte logica dall'interfaccia grafica tramite un cotroller, implementato nel particolare dalle classi LinQuedAdmin e LinQuedClient, che verranno discusse in seguito. Queste richiamano le funzioni prettamente logiche e le offrono all'interfaccia grafica, rendendo indipendenti entrambe le parti l'una dall'altra in vista di una futura modifica/estensione del codice.

2. Parte logica

Di seguito verranno elencate tutte le classi utilizzate appartenenti alla parte logica del progetto e le rispettive funzioni principali. Si è scelto di non creare la classe di puntatori smart SmartUtente poiché, almeno per le funzionalità richieste, non se ne è sentita la necessità. Il problema della memoria condivisa che un puntatore smart può potenzialmente risolvere, non è presente poiché non avverranno mai copie di uno stesso utente, dato che esso deve essere unico all'interno del database. Non vi è nemmeno il problema del garbage lasciato dalla cancellazione di un oggetto Utente (classe che verrà illustrata in seguito), poiché questa avverrà solo tramite l'amministratore e dunque gestibile in questa singola operazione richiamando il distruttore di Utente, ridefinito e dichiarato virtuale appositamente.
Tutte queste considerazioni, e quelle fatte in seguito, sono basate guardando alle funzionalità minime richieste ed al tempo limitato per la consegna del progetto. Ovviamente se il software dovesse essere successivamente ampliato si dovranno effettuare alcune modifiche al codice originale.

2.1 Profilo

La classe contiene i dati personali di un cliente. Tre campi sono di tipo stringa: nome, cognome e email e quattro di tipo lista di stringhe: titoliStudio, lingue, competenze ed esperienze. Si è scelto il contenitore STL lista poiché i campi informativi all'interno dei contenitori non andranno mai acceduti in maniera casuale, ma sempre stampati dal primo all'ultimo. L'inserimento poi avverrà sempre in coda. La classe offre i metodi per la visualizzazione della singola informazione e per la sua futura modifica e aggiornamento. Vi sono poi due categorie di funzioni che servono per la trasformazione delle liste in un unica stringa e viceversa. Le prime sono utili prevalentemente all'interfaccia grafica, quando si andranno a stampare a video le informazioni presenti nelle liste, che verranno visualizzate come un'unica stringa, separate da uno spazio. L'altra tipologia serve per il caricamento del database dal file XML (che verrà illustrato nella classe DB) poiché inserisce nella lista le informazioni quando queste sono in un unica stringa separate da spazi.

2.2 Username

La classe contiene un unico campo dati di tipo stringa: login. Essa implementa il concetto di username attraverso la quale un cliente potrà accedere alla propria sezione dedicata. Offre un metodo di visualizzazione del campo dati login e la ridefinizione degli operatori di uguaglianza e disuguaglianza che saranno utili per la ricerca di un cliente all'interno del database.

2.3 Rete

La classe modella l'insieme dei contatti che un cliente possiede. Si è scelto di avere una lista di Username e non di puntatori ad Utente, perché per le funzionalità richieste, ovvero di inserimento, rimozione e visualizzazione, il salvataggio della sola Username è sufficiente. Inoltre semplifica non di poco il caricamento del database dal file XML poiché, dato che ogni cliente avrebbe avuto puntatori verso clienti potenzialmente non ancora caricati nel database, si sarebbero dovute implementare due funzioni di load, una per il caricamento degli utenti dal file nel database e l'altra per i contatti di ogni singolo utente. La scelta del contenitore lista è stata dettata dalle stesse ragioni viste per la classe Profilo. Anche qui sono disponibili due funzioni che trasformano la lista di Username in un unica stringa di login e viceversa.

2.4 Utente

La classe Utente è una classe base polimorfa e astratta. Polimorfa poiché contiene il distruttore virtuale e astratta perché offre un metodo virtuale di ricerca di un altro utente all'interno del database puro. Esso verrà concretizzato nelle tre classi derivate Utente Basic, Utente Business ed Utente Executive, le quali differiranno l'una dall'altra solo per una diversa implementazione di questo metodo. La classe Utente contiene un campo dati Profilo, un puntatore ad una Rete di contatti e una Username. Oltre ad offrire alcune funzioni di servizio, contiene una classe annidata SearchFunctor. Questa classe costruisce un oggetto funtore, oggetto che tramite l'overloading dell'operatore operator(), può essere usato come se fosse una funzione. E' tramite questo particolare oggetto che le classi derivate effettueranno un diverso overriding della funzione di ricerca userSearch(). Se l'utente che effettua la ricerca è Basic, la funzione userSearch() istanzierà un funtore di tipo 1, il quale riempirà la lista passata per riferimento solo con i campi dati login, nome, cognome e email.

Per un utente Business istanzierà un funtore di tipo 2 e per un Executive di tipo 3, i quali avranno opzioni di accesso ai campi dati via via sempre maggiori.

2.5 DB

Il contenitore STL scelto per il salvataggio dei puntatori ad oggetti Utente è il vector. Questo perché ci sarà la necessità di accedere casualmente agli elementi del database e il vector è il contenitore che meglio ottimizza questa operazione. La classe offre i metodi di ricerca, inserimento e rimozione di un utente dal database; un metodo che restituisce una copia del campo vector che tornerà utile per stampare le informazioni di tutti gli utenti nell'interfaccia grafica; un metodo che cambia la tipologia dell'utente (Basic, Business ed Executive) che sarà utile all'amministratore. Infine sono presenti i metodi di salvataggio e caricamento del database nel e da file XML database.xml. Per queste operazioni si sono utilizzate le funzionalità offerte dalla libreria Qt.
Per il salvataggio del database nel file .xml è stato utilizzato lo XMLStreamWriter. Viene fatto associare il file allo StreamWriter, il quale scorre l'intero database per salvare, utente per utente, i vari campi dati. Viene usato il dynamic_cast per ricavare la tipologia dell'utente, la quale verrà salvata nel tag tipoutente.
Per il caricamento invece, viene utilizzato il QDomDocument. Questo crea una lista, un nodo per ogni utente con i suoi campi dati. Una volta finito di caricare tutti i tag di un nodo, viene controllata la tipologia del cliente e viene così creato un oggetto Utente, con i campi dati raccolti, di quel tipo.

3. Controller

Solamente due classi fungono da controller, collegando la parte logica a quella grafica rendendole indipendenti l'una dall'altra.

3.1 LinQuedInAdmin

La classe definisce tutte tutte le operazioni che un amministratore può effettuare nel database: la ricerca, l'inserimento e la rimozione di un cliente, la modifica della sua tipologia e il salvataggio. Tutte questa funzioni non vengono definite direttamente nella classe in esame ma richiamano metodi già visti nella classe DB. Infatti durante l'istanziazione di un oggetto LinQuedInAdmin verrà creato un oggetto DB è riempito tramite il caricamento da file XML.

3.2 LinQuedInClient

Come per la classe LinQuedAdmin, la classe LinQuedClient non fa altro che elencare tutte le possibili operazioni che un cliente può effettuare: l'aggiornamento del proprio profilo, l'inserimento e la rimozione di un contatto e la ricerca di un altro cliente all'interno del database in base alla propria tipologia. I campi dati che contiene sono un puntatore al database che verrà caricato al momento dell'istanziazione di un oggetto LinQuedInClient e un puntatore ad uno specifico utente.

4. Interfaccia grafica

Il layout grafico è stato sviluppato tramite il tool QtDesigner messo a disposizione da FrameWork QtCreator. Il comportamento invece è stato implementato tramite codice Qt che richiama le funzioni progettate nella parte logica del progetto sopraesposta. L'interfaccia grafica si divide in lato admin e lato client, fatta eccezione per la finestra principale, dalla quale si può accedere ad uno dei due lati.

4.1 MainWindow

Dato che non è stato richiesto, non si è sviluppato nessun accesso controllato per il lato admin, al quale si può accedere cliccando semplicemente sul pulsante “Login”.
Per il lato client invece si è predisposto l'accesso tramite la sola username. Se questa è presente nel database, si potrà entrare nell'interfaccia dedicata al particolare cliente accedente.

4.2 Lato amministratore

La login dell'amministratore provoca l'apertura di un nuova finestra AdminWindow. Qui troviamo tutti i widget che sono stati predisposti per le operazioni che un admin può effettuare. La pressione sul pulsante “Inserisci” avvia l'apertura di una nuova finestra AdminWindowInserisci, nella quale si possono inserire tutti i campi dati propri di un cliente, sceglierne la tipologia e così inserirlo all'interno del database. I campi Username, Nome e Cognome sono obbligatori e se non viene scelta una tipologia d'utenza esplicitamente, l'utente inserito sarà di default Basic.
E' possibile poi rimuovere un utente inserendo la sua username nella apposita label e cliccare sul pulsante “Rimuovi”.
Il pulsante “Mostra” provoca la visualizzazione dei principali campi dati di tutti gli utenti all'interno del database nella tabella sottostante. Si potrà così selezionare un cliente e cambiarne la tipologia cliccando sul pulsante “Modifica tipologia”. Se si tenta di scegliere la tipologia a cui l'utente già appartiene verrà visualizzato un messaggio di errore.
Infine tramite il pulsante “Salva database” si potranno salvare le modifiche effettuate nel file database.xml e renderle così permanenti.

4.3 Lato cliente

Dopo aver digitato la propria username nella parte client della MainWindow, si potrà accedere all'interfaccia propria del cliente. Da qui si può visualizzare il proprio profilo nella apposita finestra WindowClientProfilo cliccando sul pulsante “Mostra”, aggiornare i campi dati tramite il pulsante “Aggiorna” (tutti eccetto la Username, il Nome e il Cognome) nella sezione WindowClientAggiorna e inserire e togliere un utente dalla propria rete di contatti tramite la digitazione della sua username e la pressione dei pulsati “Inserisci” e “Rimuovi” (verrà visualizzato un messaggio di errore se l'username non è presente nel database). Se lasciati in bianco, i campi dati nella sezione WindowClientAggiorna non andranno a modificare le informazioni precedentemente inserite.
Il pulsante “Cerca” permette di ricercare un cliente all'interno del database (se presente) e di visualizzarne i dati personali nella finestra WindowClientCerca in base alla propria tipologia d'utenza, come sopraesposto.
Infine anche il cliente potrà decidere di rendere permanenti le modifiche effettuate al proprio profilo tramite il salvataggio nel file .xml.