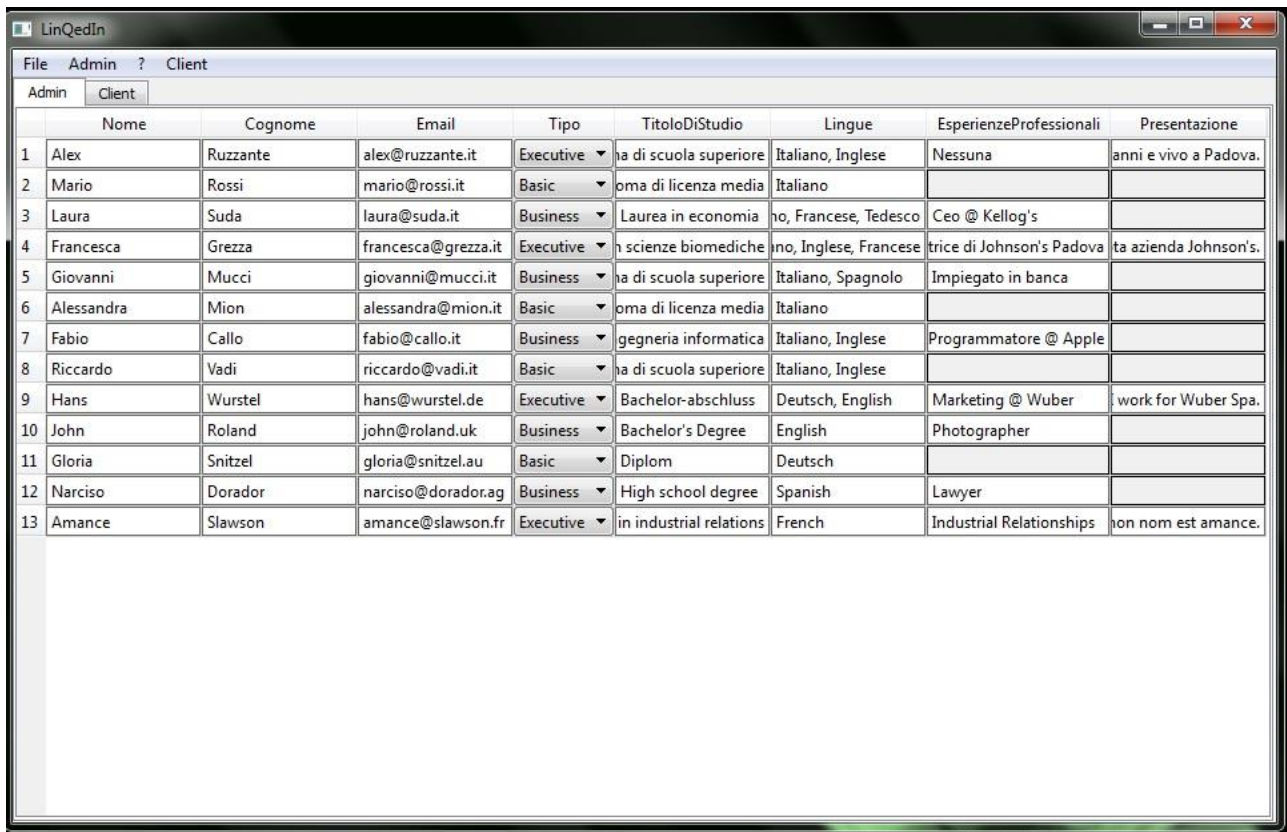


## LinQedIn

Progetto di Programmazione ad Oggetti, a.a. 2014/2015



The screenshot shows a window titled 'LinQedIn' with a menu bar (File, Admin, ?, Client) and two tabs (Admin, Client). The 'Client' tab is active, displaying a table with 13 rows of user data. The table has 8 columns: Nome, Cognome, Email, Tipo, TitoloDiStudio, Lingue, EsperienzeProfessionali, and Presentazione. Each row represents a user with various details like name, email, type (Executive, Basic, Business), degree, languages, professional experience, and a presentation note.

	Nome	Cognome	Email	Tipo	TitoloDiStudio	Lingue	EsperienzeProfessionali	Presentazione
1	Alex	Ruzzante	alex@ruzzante.it	Executive	ha di scuola superiore	Italiano, Inglese	Nessuna	anni e vivo a Padova.
2	Mario	Rossi	mario@rossi.it	Basic	oma di licenza media	Italiano		
3	Laura	Suda	laura@suda.it	Business	Laurea in economia	no, Francese, Tedesco	Ceo @ Kellog's	
4	Francesca	Grezza	francesca@grezza.it	Executive	n scienze biomediche	ino, Inglese, Francese	trice di Johnson's Padova	ta azienda Johnson's.
5	Giovanni	Mucci	giovanni@mucci.it	Business	na di scuola superiore	Italiano, Spagnolo	Impiegato in banca	
6	Alessandra	Mion	alessandra@mion.it	Basic	oma di licenza media	Italiano		
7	Fabio	Callo	fabio@callo.it	Business	gegneria informatica	Italiano, Inglese	Programmatore @ Apple	
8	Riccardo	Vadi	riccardo@vadi.it	Basic	na di scuola superiore	Italiano, Inglese		
9	Hans	Wurstel	hans@wurstel.de	Executive	Bachelor-abschluss	Deutsch, English	Marketing @ Wuber	work for Wuber Spa.
10	John	Roland	john@roland.uk	Business	Bachelor's Degree	English	Photographer	
11	Gloria	Snitzel	gloria@snitzel.au	Basic	Diplom	Deutsch		
12	Narciso	Dorador	narciso@dorador.ag	Business	High school degree	Spanish	Lawyer	
13	Amance	Slawson	amance@slawson.fr	Executive	in industrial relations	French	Industrial Relationships	on nom est amance.

Sistema Operativo di Sviluppo: Windows 7 Ultimate 64bit

Versione compilatore: MinGW 4.8.2 32bit

Versione Qt: 5.3

### Introduzione

Il progetto è stato sviluppato in ambiente Windows mediante l'editor Qt Creator, senza l'utilizzo di Qt Designer. In allegato è presente il file "*progetto.pro*" che permette la compilazione nelle macchine del laboratorio Paolotti. Inoltre tra i file consegnati vi è il file "*prova.xml*" che consente di popolare rapidamente il database.

### Principali Scelte Progettuali

La parte logica comprende la gerarchia di classi: Utente(base astratta polimorfa), UtenteBasic, UtenteBusiness, UtenteExecutive.

UtenteExecutive è derivata da UtenteBusiness che deriva da UtenteBasic che deriva da Utente. Le 3 classi differiscono per il numero di campi dati crescente, infatti un UtenteBusiness può inserire più informazioni rispetto ad un Basic e così via. La classe Utente è resa astratta dal metodo virtuale puro clone() e possiede distruttore marcato virtuale.

Il contenitore scelto per il Database è un vector ed in esso vengono effettuati inserimenti e rimozioni in coda, anche per i contatti è stato scelto un vector con aggiunte e rimozioni esclusivamente mediante push\_back() e pop\_back().

La finestra principale del programma si compone di due schede: Admin e Client.

Admin permette la gestione vera e propria degli utenti di LinQedIn con una visualizzazione tabellare, mediante aggiunte/rimozioni/modifiche ed una semplice ricerca per email.

Client permette ad un utente presente nel database, dopo aver effettuato il login, di modificare i propri dati, visualizzare i propri contatti, aggiungere/rimuovere un contatto, cercare altri utenti e visualizzare il risultato in base alla propria tipologia di account.

Un utente Executive può, ad esempio, vedere la lista contatti di un utente che è presente tra i propri contatti mediante una ricerca per email. Cosa che un utente Business non riesce a fare.

L'unico operatore di RTTI utilizzato è `dynamic_cast`, ed è stato utilizzato per individuare se gli elementi di tipo `Utente*` memorizzati nel database fossero effettivamente di tipo `Basic`, `Business` o `Executive`.

Il caricamento/salvataggio dei dati viene effettuato su file `.xml` sfruttando `QXmlStreamReader` e `QXmlStreamWriter` rese disponibili dalla libreria Qt.

La gestione della memoria è principalmente affidata a Qt, fatta eccezione per la classe `Table` che ridefinisce il distruttore e si occupa della distruzione dell'oggetto puntato dal puntatore `Utente*` presente in ogni riga della tabella tramite il distruttore virtuale.