Relazione Progetto Programmazione ad Oggetti

Nome: Follador Saverio
Matricola: 1096984

Indice

1 Introduzione **3**
1.1 Ambiente di sviluppo 3
1.2 Tempo di sviluppo 3
1.3 Compilazione ed esecuzione 3
1.4 Descrizione 3

2 Struttura **4**
2.1 Informazioni generali 4
2.2 Classi modellate 4
2.2.1 Utente 4
2.2.2 Prodotto 4
2.2.3 Database 4
2.3 Salvataggio su file 4
2.4 Divisione Grafica-Logica 5
2.4.1 Controller 5
2.4.2 Interfaccia 5

1 Introduzione

1.1 Ambiente di sviluppo

- **Sistema Operativo:** Manjaro Linux 16.10
- **Compilatore:** GCC 6.3.1
- **Versione Qt Creator:** 4.2.1
- **Versione Qt:** 5.7.1

1.2 Tempo di sviluppo

- **Progettazione modello e GUI:** 2.5 ore
- **Codifica modello e GUI:** circa 45 ore
- **Debugging:** 5 ore
- **Testing:** 4 ore

1.3 Compilazione ed esecuzione

Per compilare il progetto posizionarsi tramite terminale all'interno della cartella dello stesso. A questo punto, eseguire il comando **qmake** per generare il makefile. Eseguire quindi il comando **make** per avviare la compilazione. A compilazione terminata all'interno della cartella si potrà trovare un file eseguibile **ProgettoP2**.
Nella cartella *database_esempio* sono disponibili due database già popolati (**databaseUtenti.txt** e **databaseProdotti.txt**). Per utilizzarli, li si dovrà copiare nella cartella in cui si è compilato il progetto; in caso di mancanza questi due file si creeranno automaticamente al primo accesso al pannello amministratore di utente per *databaseUtenti.txt* e di prodotto per *databaseProdotti.txt*. Le credenziali di accesso per l'amministratore sono (username,password)=(admin,admin).

1.4 Descrizione

Come da consegna si è sviluppato un software che permettesse l'accesso da parte di alcune tipologie di utenti ad un database. Il database che si è scelto di modellare è un database di informazioni sui prodotti di una qualsiasi azienda. Al database si può accedere come utente o come amministratore. Esistono tre tipologie di utenti (*casuale*, *utilizzatore* e *rivenditore*), ciascuna delle quali può ricercare prodotti e visualizzarne alcune informazioni. Le informazioni visualizzate saranno più o meno complete a seconda della tipologia di utente. La ricerca riguarda più o meno caratteristiche a seconda del tipo di utente che la effettua. L'amministratore può inserire, eliminare o modificare prodotti e utenti.

2 Struttura

2.1 Informazioni generali

Per modellare i database di Utenti e Prodotti si è utilizzata la struttura dati **vector<T>** contenuta nella libreria STL. Entrambi i database sono formati da un vector di puntatori a oggetti del tipo rispettivo. Non sono stati utilizzati puntatori smart in quanto, data la tipologia di database modellati, non vengono mai effettuate copie di oggetti. La gestione del garbage è affidata alle singole funzioni di eliminazione.

2.2 Classi modellate

2.2.1 Utente

Gli utenti a cui viene garantito l'accesso al database sono gestiti da una gerarchia di classi. Dalla classe base astratta **Utente** derivano le classi **UtenteCasuale**, **UtenteUtilizzatore** e **UtenteRivenditore**. *UtenteCasuale* rappresenta un utente che si presuppone acceda saltuariamente al database dell'azienda. Pertanto ha accesso solo ad alcune informazioni dei prodotti con le sue ricerche (nome e uso del prodotto). *UtenteUtilizzatore* rappresenta un utente che utilizza i prodotti dell'azienda. In quanto utilizzatore ha accesso anche alla durata dei prodotti oltre che a nome e uso. *UtenteRivenditore* rappresenta un rivenditore dei prodotti dell'azienda. Ha accesso a nome, uso, durata e prezzo dei prodotti.
La classe base Utente contiene le informazioni base di ciascun utente; le immagazzina attraverso oggetti di due classi create ad hoc: **LoginPw** e **Info**.
I diversi privilegi di ricerca per i vari tipi di utenti sono implementati tramite funtori (oggetti della classe **Funtore**).

2.2.2 Prodotto

I prodotti sono gestiti da una classe **Prodotto**. Ogni oggetto *Prodotto* contiene degli attributi che ne esprimono le informazioni da visualizzare e dei metodi di *set* e *get*.

2.2.3 Database

I database sono modellati tramite due classi: **DatabaseProdotti** e **DatabaseUtenti**. Salvo alcune piccole variazioni sono due classi speculari, contententi nella parte privata il contenitore vector di puntatori e nella parte pubblica i metodi per la gestione del database. La classe *DatabaseUtenti* contiene inoltre un metodo per l'autenticazione dell'utente.

2.3 Salvataggio su file

Si è deciso di mantere il salvataggio su file quanto più semplice possibile. Il salvataggio avviene tramite stampa su file .txt riga per riga degli attributi di ciascun oggetto Prodotto o Utente. Eventuali attributi vuoti corrispondo a righe vuote.
Il caricamento avviene pertanto scorrendo riga per riga il file e creando mano a

mano gli oggetti.
Ogni qualvolta il database viene modificato il file di testo corrispondente viene aggiornato.

2.4 Divisione Grafica-Logica

Si è cercato di seguire l'architettura **MVC** (Model-View-Controller). La parte grafica è separata dalla parte logica; entrambe vengono messe in comunicazione fra loro tramite il controller.
Ciascuna schermata del progetto è identificata da una specifica classe che ha il compito di costruirne il layout e di creare le connect necessarie. Ciascuna di queste classi ha degli slot personalizzati, associati alle varie funzionalità del pannello in questione. Salvo alcuni casi in cui non è necessario, gli slot interagiscono con un oggetto controller.
Lo sviluppo dell'interfaccia grafica è stato effettuato interamente "a mano", senza utilizzare tool del Framework.

2.4.1 Controller

Il controller si occupa di collegare la parte logica e la parte grafica. Sono presenti due classi a questo scopo: **ControllerAdmin** e **ControllerUtente**. Un oggetto controller viene creato ogni qualvolta si effettua l'accesso. *ControllerAdmin* possiede metodi per inserire, modificare ed eliminare oggetti di entrambi i database; *ControllerUtente* possiede metodi solo per interrogare il database dei prodotti.

2.4.2 Interfaccia

L'intera GUI del progetto è stata realizzata con codice scritto "a mano", senza l'utilizzo di tools del Framework come *Qt Designer*. Questa scelta è dovuta alla necessità di dover comunque ricontrollare tutto il codice creato da Qt Designer, aumentando considerevolmente le ore necessarie (con il rischio di superare il monte ore stabilito dalla consegna). Inoltre, vista la scarsa complessità della GUI modellata, lo scrivere il codice interamente a mano si è rivelato un compito "meccanico" e per nulla lento.
Per ogni schermata del progetto è stata sviluppata una classe apposita. Ciò ha permesso di semplificare il codice necessario e di separarlo da quello di funzionalità differenti (es. schermate Utente e Admin). Tutte le classi create per la GUI sono derivate dalla classe base **QWidget**. Le varie schermate utilizzano un oggetto locale di tipo **QGridLayout** per disporre in maniera ordinata i vari oggetti all'interno della schermata.