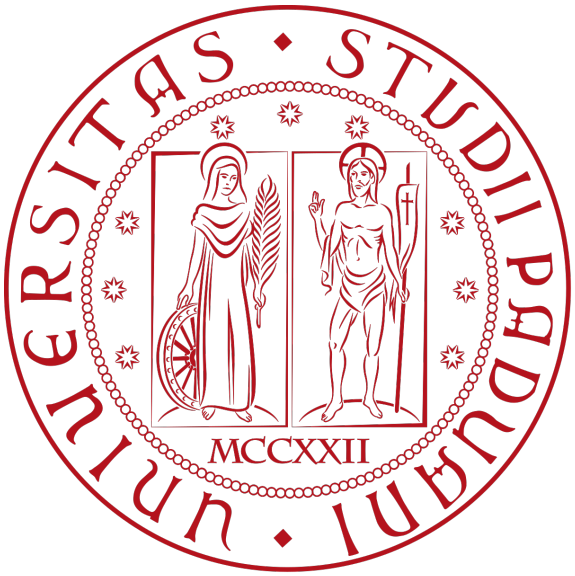

Programmazione ad Oggetti
a.a. 2022/2023

Password Manager

2010004 - FRANCESCO CONSTANTINO BULYCHOV

Indice

1 Introduzione

Password Manager é un software per la gestione di informazioni sensibili che offre un'interfaccia intuitiva e semplice da usare. Gli utenti possono creare, modificare, eliminare e visualizzare diversi tipi di dati, tra cui Login per siti e applicazioni, Carte di credito/debito e Note contenenti informazioni sensibili come chiavi private SSH o codici di accesso a wallet di criptovalute. Tutte le informazioni vengono alla fine salvate in un file criptato che impedisce l'accesso a quest'ultime al di fuori del software e ne garantisce la riservatezza.

2 Descrizione del modello

Il modello parte da una classe astratta *AbstractItem* che rappresenta le informazioni comuni a tutti gli oggetti, ovvero identificatore univoco e nome. Per il nome vengono forniti i rispettivi metodi *getter* e *setter*, mentre per l'identificatore solamente il *getter* in quanto viene sempre generato insieme all'oggetto e mai modificato.

La classe concreta *Login* rappresenta l'insieme delle informazioni che vengono solitamente richieste per l'accesso ad un sito o un'applicazione. La classe concreta *Card* mette insieme tutti i dati riguardanti una carta di credito o di debito, come puó essere il numero, la data di scandenza, il nome del titolare e il codice sul retro. Infine con la classe concreta *Note* si vuole lasciare la libertá all'utente di rendere privato un qualsiasi testo.

Per tutto il codice del programma viene utilizzato abbondantemente il design pattern *Visitor*, e a questo scopo sono state create le classi astratte *IVisitor* e *IConstVisitor*, le quali differiscono unicamente per il fatto che la seconda lascia necessariamente inalterato l'oggetto visitato. Di conseguenza, in *AbstractItem* sono stati inseriti i metodi virtuali puri *accept* per accettare i due tipi di *Visitor*, poi implementati nelle varie classi concrete.

Il container utilizzato é una lista unicamente linkata dove il campo dati é un puntatore ad un *AbstractItem*. Al suo interno vi é anche una classe *Iterator* che rende piú semplice interagire con la lista. Oltre ai metodi piú basilari sono anche presenti metodi per inserire e rimuovere oggetti, per ritornarne uno specifico o per creare un nuovo container contenente solo un sottogruppo di elementi. É presente anche un metodo che genera un identificatore unico controllando tutti gli oggetti all'interno del container, utilizzato durante la creazione di nuovi oggetti.

3 Polimorfismo

L'utilizzo principale del polimorfismo riguarda il *design pattern Visitor* nella gerarchia *AbstractItem*, il quale é stato abbondandemente sfruttato in varie parti del codice grazie alle classi *IVisitor* e *IConstVisitor*.

In breve, le parti principali in cui il Visitor é stato utile sono state per la lettura e scrittura sul file XML, per la visualizzazione e la modifica dell'oggetto selezionato, per l'aggiunta di un nuovo oggetto.

Siccome ogni elemento ha dei campi dati unici, la scelta di dividere tutto in classi Visitor specifiche ha reso il codice piú organizzato, nonché renderá piú semplice un'eventuale aggiunta di elementi.

Un altro esempio di polimorfismo utilizzato all'interno del programma é il metodo *clone* nella gerarchia *AbstractItem*, il quale é stato utile all'interno del container per creare un costruttore di copia profondo, l'operatore di uguaglianza profondo e il metodo *searchByName*, che per l'appunto ritorna un container contenente un sottogruppo di oggetti specifici senza modificare il container originale. É stato necessario intraprendere questa soluzione in quanto il campo dati della lista unicamente linkata non é un oggetto bensí un puntatore ad un oggetto, e il metodo clone é stato il modo piú semplice per creare una copia di quest'ultimo.

4 Persistenza dei dati

La persistenza dei dati viene gestita mediante l'uso di un file XML. Ogni oggetto all'interno del file viene racchiuso sotto forma di Item, salvando il tipo dell'oggetto in chiaro e tutte le restanti informazioni in modo cifrato. Entrando nel dettaglio, ogni stringa contenente dati sensibili é stata inizialmente cifrata utilizzando l'algoritmo di cifratura AES-256, dopodiché é stata convertita in base64 per permettere la compatibilitá di caratteri tra QString e AES-256. Le librerie aggiuntive usate a questo scopo sono state la libreria base64 di René Nyffenegger e la libreria openSSL per l'algoritmo di cifratura. Un'altra nota riguardante la persistenza dei dati riguarda l'identificatore univoco degli oggetti, che non viene memorizzato bensí generato nuovamente ad ogni nuova lettura del file, mantenendone ovviamente la sua unicitá.

5 Funzionalitá implementate

Funzionalitá generali:

- gestione di tre tipologie di oggetti
- conversione e salvataggio in formato XML utlizzando cifratura AES-256
- funzionalità di ricerca in tempo reale basata sulla presenza della stringa data all'interno del nome dell'oggetto
- messaggio di errore in caso di problemi durante la lettura di un file
- pulsanti "Save", "Save As", "New Item" disabilitati in caso di errori durante la lettura del file
- utilizzo di espressioni regolari per il corretto inserimento della data nell'oggetto "Card"

Funzionalitá grafiche:

- barra dei menù in alto
- utilizzo di una toolbar con le rispettive icone
- scorciatoie da tastiera (mostrate anche nelle voci del menù)
- gestione del ridimensionamento
- utilizzo di icone nei pulsanti
- utilizzo di icone accanto gli oggetti per identificarne il rispettivo tipo
- pulsante per mostrare o nascondere le password

6 Rendicontazione ore

Attivitá	**Ore Previste**	**Ore Effettive**
Studio e progettazione	10	4
Sviluppo del codice del modello	10	10
Studio del framework Qt	10	12
Sviluppo del codice della GUI	10	17
Test e debug	5	3
Stesura della relazione	5	3
Totale	50	46

Lo studio e la progettazione sono stati particolarmente brevi in quanto il tema del progetto era stato deciso molto tempo in anticipo, mentre l'estetica e le caratteristiche principali sono state ispirate da password manager giá esistenti.

Anche lo sviluppo del codice del modello non ha richiesto molto tempo in quanto é stato tutto abbondantemente trattato durante il corso e avevo giá familiaritá con l'argomento. Sicuramente la parte piú lunga é stata l'implementazione della persistenza dei dati e la cifratura delle informazioni.

Lo studio del framework Qt é stato composto da un breve studio della documentazione e da un'ampia analisi di progetti giá esistenti, nel tentativo di comprenderne le meccaniche e di entrare nella logica di Qt prima di iniziare a scrivere il codice.

Lo sviluppo del codice della GUI é stata sicuramente la parte piú lunga proprio per la poca familiaritá con il framework. Con il passare delle ore é stato tutto sempre piú semplice.