UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO DI MATEMATICA

LAUREA TRIENNALE IN INFORMATICA

PROGETTO DI PROGRAMMAZIONE AD OGGETTI

Libreria di Contenuti Digitali di

FantaScienza

Lorenzo Soligo 2101057, Antonio Tang 2111017 Relazione di Soligo Lorenzo

Contents

1 Introduzione

La ScienceFiction Library è una libreria di contenuti di fantascienza. E' possibile gestire una vasta gamma di tipologia dei contenuti, come libri, fumetti, film, serie tv e videogiochi. Questi possono essere creati, modificati e cancellati dall'utente, che potrà anche decidere di importare una libreria già fatta in formato Xml o Json e di esportare a sua volta la propria libreria in uno dei due formati.
La libreria è pensata per essere utilizzata da un singolo utente, che potrà gestire i propri contenuti in modo semplice e veloce. Ed è stata realizzata per poter essere percepita come uno strumento flessibile e familiare nell'nterfaccia. I principali punti forti sono la possibilità di filtrare i contenuti per cercare solo quelli che ci interessano, combinando diversi paramentri per una ricerca granulare. Ogni tipologia di contenuto è poi creata in modo da mostrare tutte le informazioni più importanti, che possono variare a seconda del tipo di contenuto, caratteristica presente anche nel momento in cui si decide di modificarne uno, in modo da riuscire a descrivere nella maniera più completa possibile il contenuto. La vista generale invece da un'idea d'insieme della libreria, mosrando i contenuti con le relative copertine e un frame personalizzato: se un contenuto è stato già Visto (Watched) il frame sarà di un colore verde neon, se è tra i Preferiti (Starred) allora sarà di colore giallo fluo e se presenta entrambi queste caratteristiche allora sarà evidenziato con un colore viola neon. Tutto questo per rendere più facilmente riconoscibili tra loro i contenuti e per rendere l'esperienza di utilizzo più gradevole.
Tutte le azioni implementate sono accessibili anche via shortcut da tastiera per rendere l'utilizzo più rapido.
L'intero progetto è disponibile su GitHub(https://github.com/Solgio/OOProject) e analizzato staticamente con SonarQube. La scelta di limitare la libreria a solo elementi fantascientifici è guidata da gusto personale e volonta di presentare un progetto particolare.

2 Descrizione del Modello

Il progetto è stato realizzato cercando di dividere il più possibile la parte logica da quella di gestione dell'UI. Il grafico UML qui sotto mostra le classi principali della parte logica.

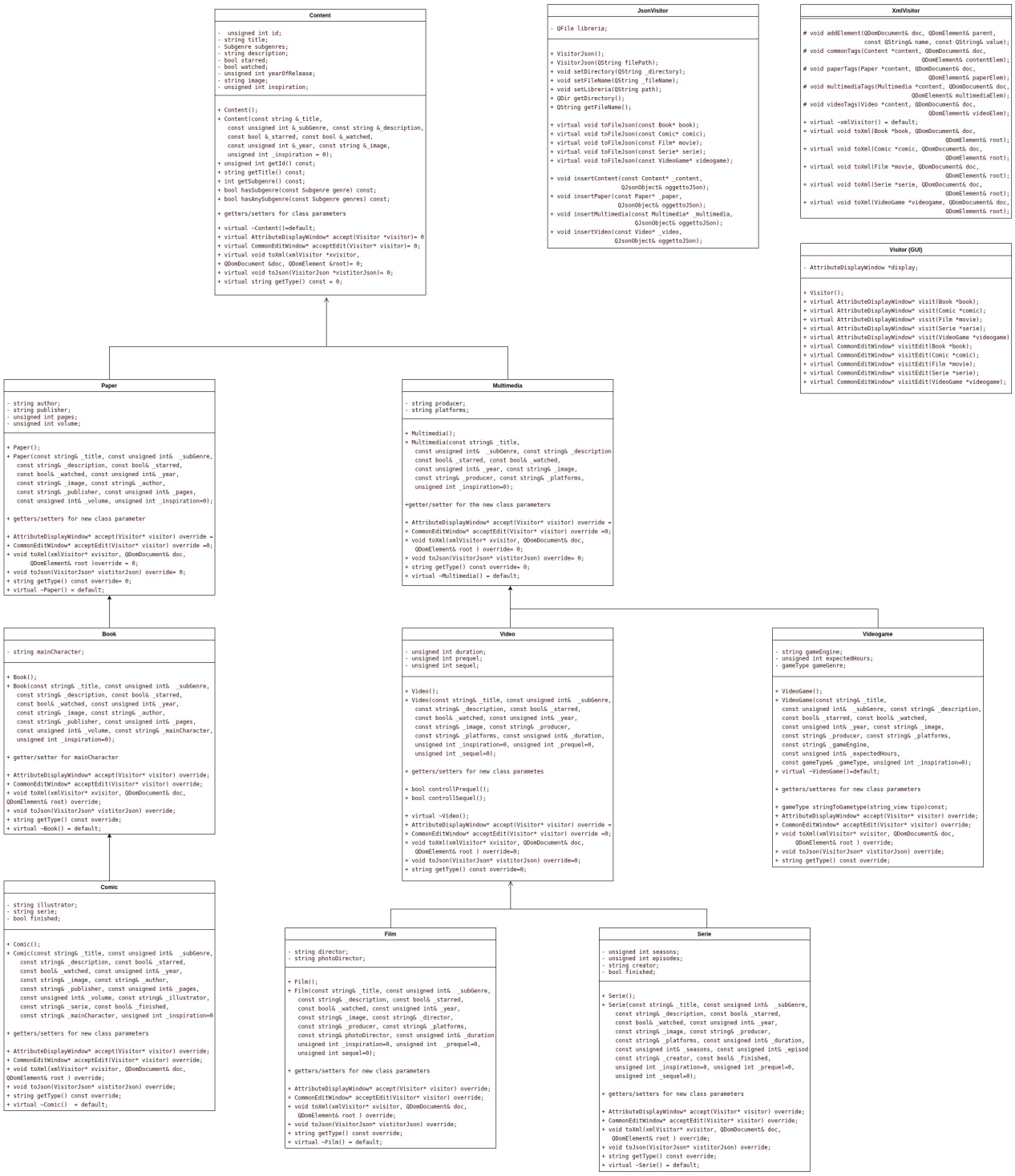

Figure 1: Diagramma UML delle classi principali della parte logica

La libreria infatti è composta da elementi appartenenti alla classe astratta `Content` che ha come parametri e metodi quelli comuni a tutti i contenuti, per esempio titolo, anno di pubblicazione. Inoltre ogni contenuto possiede un Id unico che è necessario per paramentri come *Inspiration*, o *Prequel/Sequel*, che rappresentano relazioni tra Content che però non sarebbe possibile rappresentare diversamente una volta che questi devono essere salvati nei formati dati implementati. L'idea è quella di un *"puntatore"* a un altro Content, che permetta di mantenere

la relazione nei dati rappresentanti i contenuti.
Particolare attenzione è stata posta alla gestione del parametro *Subgenre* che corrisponde ai diversi sottogeneri della fantascienza che un contenuto può avere. La particolarità che si è voluto ricercare è il fatto che un contenuto può avere diversi sottogeneri e quindi il parametro è stato modellato come una *Bitmask* in modo da poter rappresentare tutte le combinazioni in modo efficiente e compatto. Ovviamente, altre alle funzioni standard per la gestione di parametri (getters/setters) è stato necessario anche la costruzione di:

- `getSubgenreString` che ritorni le stringe con i nomi dei sottogeneri separati da una virgola.
- `hasSubgenre` che controlla se un content ha determinati sottogeneri, sfruttato in seguito per il filtraggio dei contenuti.

Come derivazioni di Content sono state imeplementate le classi astratte `Paper` e `Multimedia` che rappresentano rispettivamente le classi di tutti i contenuti cartacei e di tutti i contenuti multimediali, per esempio video+audio o video+audio+interazione.
Da Paper sono state derivate quindi le classi `Book` e `Comic`, che rappresentano rispettivamente i libri e i fumetti, mentre da Multimedia sono state derivate le classi `Video` e `VideoGame`, che rappresentano rispettivamente i contenuti video come film e serie tv e i videogiochi.
Come anticipato sopra anche in Video, e quindi nelle derivate `Film` e `Serie`, la gestione *Sequel e Prequel* è stata implementata sfruttando gli Id dei Content. Inoltre è stato mantenuto il vincolo che se un contenuto A è prequel di un contenuto B allora B è sequel di A. Quindi non solo viene effetuato un controllo in modo che questo vincolo sia mantenuto ma quando viene distrutto un *Film o Serie* se ha un sequel o prequel allora viene cercato all'interno della lista e il parametro viene posto a 0, in modo da non avere riferimenti a elementi non più presenti all'interno della libreria.
Per la gestione dinamica dell'interfaccia grafica è stato implementato il design pattern Visitor, che permette di mostrare le informazioni specifiche di ogni Content in modo dinamico, sia nella vista generale che nelle finestre di dettaglio e modifica. Stesso approccio è stato sfruttato anche per la persistenza dei dati in Xml e Json.
In generale l'obiettivo della struttura gerarchica così costruita era quelle di avere uno scheletro facilmente estendibile e modificabile, in cui poter aggiungere nuovi tipi di Content senza dover modificare le classi già esistenti, ma semplicemente creando una nuova classe che estende Content o le derivate.

3 Polimorfismo

L'utilizzo principale del polimorfismo riguarda il design pattern *Visitor*, nella gerarchia di Content. Esso è utilizzato per due motivi principali:

- Per poter implementare il salvataggio a Json con *toJson* e a XML con *toXml*, in modo che possano essere salvati nel modo corretto in base al tipo di Content.
- Per poter gestire in modo dinamico l'interfaccia grafica di *DetailWindow* con `accept` e *EditWindow* con `acceptEdit`, in modo che in base al tipo di Content vengano mostrati solo i campi proprio di quel tipo.

3.1 GUI Visitor

Le rese grafiche degli dei Content sono 3:

- *ContentCard*, che mostra una card del Content nella vista generale
- *DetailWindow*, che mostra i dettagli del Content selezionato
- *EditWindow*, che permette di modificare il Content selezionato

La prima per natura non richiede polimorfismo in quanto è una card generica che mostra solo la Copertina e il Titolo del Content, mentre le altre due richiedono il polimorfismo per poter mostrare i campi specifici del Content selezionato.
Per queste due infatti è stato implementato il design pattern Visitor che costruisce di conseguenza la pagina di dettaglio e di modifica del Content selezionato tra quelle implementate, una per tipo di Content.
La vista è costruita in modo da poter mostrare i campi specifici del tipo di Content selezionato, in modo da poter mostrare tutte le informazioni più importanti, che possono variare a seconda del tipo di Content, caratteristica presente anche nel momento in cui si decide di modificarne uno, in modo da riuscire a descrivere nella maniera più completa possibile il Content.

4 Persistenza dei Dati

Per la persistenza dei dati sono stati implementati due metodi, uno per il salvataggio in formato Json e uno per il salvataggio in formato Xml.
Questi metodi utilizzano il design pattern Visitor, in modo da poter salvare i contenuti in modo corretto in base al tipo di Content e in modo semplice grazie all'utilizzo di funzioni .
Tutto questo è gestito dalla classe *Library* con la funzione `saveLibrary`, che si occupa di chiamare il metodo di salvataggio corretto in base al formato scelto dall'utente che una volta scelto il file, se presente, chiama la funzione polimorfa `toXml` o `toJson` che ne gestisce il salvataggio.
La funzione toXml sfrutta gli strumenti di Qt per la serializzazione in Xml ed è strutturata in modo modulare per evitare codice inutile e per una gestione che risulta più leggibile. La gestione dei tipi è gestita con l'inserimento di un tag che possono essere `Book`, `Comic`, `Film`, `Serie` o `VideoGame` a seconda del tipo di Content che si sta serializzando.
Similmente la funzione toJson sfrutta le librerie di Qt per la serializzazione in Json aggiungendo però una gestione degli errori in caso il file Json non sia valido, in modo da evitare crash dell'applicazione. Inoltre gli oggetti vengono salvati raggruppando tra loro i contenuti dello stesso tipo.
Per finire, se durante l'utilizzo vengono fatte operazioni che modificano lo stato della libreria (Import, Add, Delete, Edit) e tali cambiamenti non sono salvati su un file, una finestra di dialogo avviserà l'utente e chiederà se annullare la chiusura, chiudere senza salvare o salvare in formato Json.

5 Funzionalità

Per quanto riguarda il modello logico la principale funzionalità, oltre alla sovracitata gestione dei Subgenre, è la scelta di implementare il design pattern del singleton in modo da poter creare una solo istanza della libreria, senza copie inutili e migliorando la sicurezza, anche grazie all'utilizzo degli `unique_ptr`.
Prima di tutto un'ampia gamma di shortcut da tastiera per compiere tutte le azioni principali senza uso obbligatorio del mouse.

Shortcut	Azione
MAIN WINDOW	
CTRL+N	Crea un nuovo oggetto
CTRL+I	Importa una libreria da file
CTRL+S	Salva la libreria su file Json
CTRL+SHIFT+S	Salva la libreria su file Xml
CTRL+F	Apri/Chiudi il menù filtri
CTRL+SHIFT+F	Ripulisci i filtri
CTRL+K	Focus su barra di ricerca
F5	Aggiorna la vista
CTRL+D	Cambia la direzione di ordinamento
DETAIL e EDIT WINDOW	
ESC	Chiude la finestra corrente
CTRL+S	Salva le modifiche al Content
CTRL+Z	Annulla le modifiche al Content

Table 1: Tabella delle Shortcut implementate

Come si può vedere le funzionalità sono diverse. In particolare il filtro permette la ricerca di elementi filtrando in modo da mostrare solo e unicamente quei contenuti che rispettano i parametri di ricerca che vengono evidenziati in verde fluo se selezionati.
Vi è un conteggio degli elementi che rispettano i filtri e che sono visibili nella vista generale, in modo da poter avere un'idea di quanti contenuti sono stati filtrati e il pulsante che nasconde i filtri viene evidenziato se i filtri sono attivi. Ovviamente si è cercato di associare ad ogni bottone di azione una icona che ne rappresentasse l'azione, in modo da rendere l'interfaccia più intuitiva e familiare.
Per godere al meglio dell'interfaccia è stato inoltre applicato un sistema automatico di riadattamento dell'organizzazione della LibraryWindow in base alla grandezza della schermata, grandezza della sezione di filtro e grandezza della barra di ricerca, in modo da poter visualizzare il maggior numero di contenuti possibile.
All'interno dell'edit Window tutti i parametri possono essere modificati. Particolare attenzione hanno avuto i campi che riguardano *Inspiration* e, se presenti, *Prequel e Sequel* che invece che mostrare l'Id, mostrano i titolo dei Content disponibili.
Per quanto riguarda la scelta dei colori nella mainWindow, le card dei Content sono di colore Viola se sono stati sia segnati come Visti sia segnati come Preferiti, di colore Verde se sono stati Visti e di colore Giallo se sono stati segnati come Preferiti. Per godere dello stile per come è stato pensato si consiglia una risoluzione di 1920x1080, per evitare deformazioni o finestre tagliate.

Infine la sopracitata gestione degli Unsaved Changes, che permette di salvare i cambiamenti non ancora salvati su file, in modo da evitare perdite di dati in caso di chiusura accidentale dell'applicazione, e la speciale gestione del parametro Subgenre.
Vi sarebbero inoltre delle ulteriori funzionalità che sarebbe possibile inserire ma che per limiti di tempo non abbiamo implementato al momento della consegna, tra tutti:

- Possibilità di cancellare i contenuti direttamente dalla mainWindow, selezionandoli.
- Possibilità di aggiungere alla libreria caricata, un'altra libreria, fondendole (al momento vengono sovrascritte)
- Poter cambiare il tipo di Content di un elemento, per esempio da Film a Serie o da Libro a Fumetto, in modo da poter cambiare il tipo di Content senza doverlo cancellare e ricreare.
- Possibilità di cambiare da modalità scura, unica disponibile al momento, a modalità chiara, in modo da poter scegliere l'interfaccia più adatta ai propri gusti.

6 Rendiconto Ore di Lavoro

In questa sezione viene presentato il rendiconto delle ore di lavoro impiegate per lo sviluppo del progetto. Le ore sono state suddivise in base alle attività svolte, come la progettazione, la scrittura del codice e le revisioni.
Tale conteggio di ore è limitato al mio operato sul progetto e non tiene conto delle ore di lavoro del mio compagno.

Attività	**Ore Previste**	**Ore effettive**
Studio e Progettazione Logica	15	15
Sviluppo codice del modello	20	20
Sviluppo del codice della GUI	10	25
Sviluppo del codice di Filtraggio e Ricerca	10	15
Scelte stilistiche e di UX	0	5/10
Test e Debug	10	15
SonarQube review	5	5
	Totale	100/105

Table 2: Tabella delle Ore di Lavoro

Il monte ore totale è significamente più alto rispetto a quello previsto. Ciò è dovuto a diversi fattori. In primo luogo c'è stata una certa ricerca per la realizzazione di un'interfaccia grafica che non fosse solo pienamente funzionante e funzionale ma che potesse essere anche gradevole

e familiare per l'utente, che ha portato a un aumento del tempo di sviluppo.
In secondo luogo, abbiamo scelto di implementare un sistema complesso come quello di Subgenre, che ha richiesto un notevole sforzo di progettazione e implementazione, che però si è rivelato utile per la controparte di ricerca e filtraggio dei contenuti, che sono stati implementati in modo da essere flessibili e facili da usare.
Infine, il progetto è stato sviluppato con l'ausilio di una strumento di analisi statica del codice, SonarQube, che ha permesso di correggere delle convenzioni, possibili miglioramenti al codice e ci ha aiutato a comprendere come scrivere codice migliore. La suddivisione dei compiti è stata fatta in modo da poter completare il maggior numero di funzionalità possibili rispetto a quello che potevamo immaginare in un progetto di questo tipo, tenendo conto dei diversi approcci al progetto e allo sviluppo del codice, come esmplificato dalle statistiche di GitHub.
Segue il rendiconto delle implementazioni sviluppate.

6.1 Implementazioni di Lorenzo Soligo

- MODELLO LOGICO
  - Creazione modello Logico da Content fino a Video escluso VideoGame
  - Persistenza dei dati Xml
- GUI
  - Creazione della finestra principale e delle sue funzionalità
  - Modello iniziale di Detail e Edit
  - Sistema di filtraggio e ricerca dei contenuti
  - Ricerca delle Icone e Impostazione grafica
  - Shortcut da tastiera

6.2 Implementazioni di Antonio Tang

- MODELLO LOGICO
  - Creazione modello Logico da Video fino a Serie incluso VideoGame
  - Persistenza dei dati Json
- GUI

– Creazione della finestra Detail e Edit e delle loro funzionalità

– Creazione delle Pagine specifiche per tipo di Detail e Edit