Programmazione ad oggetti
Progetto “Semilibertà”

Todescato Matteo

Matricola 1121857

Anno 2016/2017

Indice

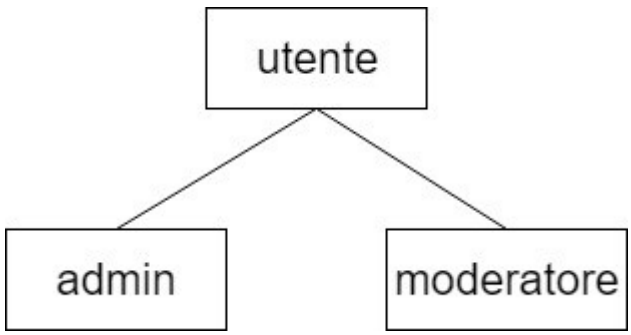
1 Scopo del progetto

Il progetto si ha lo scopo di creare un'applicazione per raccogliere informazioni riguardanti pubblicazioni di vario tipo, concedendo agli utenti diversi privilegi nella modifica della raccolta.
Ci sono tre tipi di utenti: Amministratore che hanno il controllo completo sulla lista utenti e pubblicazioni, Moderatore che possono compiere azioni solo sulle pubblicazioni e utente che può accedere in sola lettura alla raccolta delle pubblicazioni.
Le pubblicazioni possono essere di tre tipi; pubblicazioni online se non presenti in forma cartacea, libri e articoli su riviste che possono essere gestite come scritto sopra da amministratori e moderatori.
Si vuole fornire funzioni ordinamento e ricerca in tempo reale per le pubblicazioni fornendo dei filtri per la ricerca e per l'ordinamento.

2 Descrizione delle gerarchie di tipi usate

Il progetto contiene tre gerarchie, due appartengono al modello logico e sono le gerarchie degli utenti e delle pubblicazioni, la terza appartiene alla GUI e serve per visualizzare, modificare ed inserire nuove pubblicazioni.

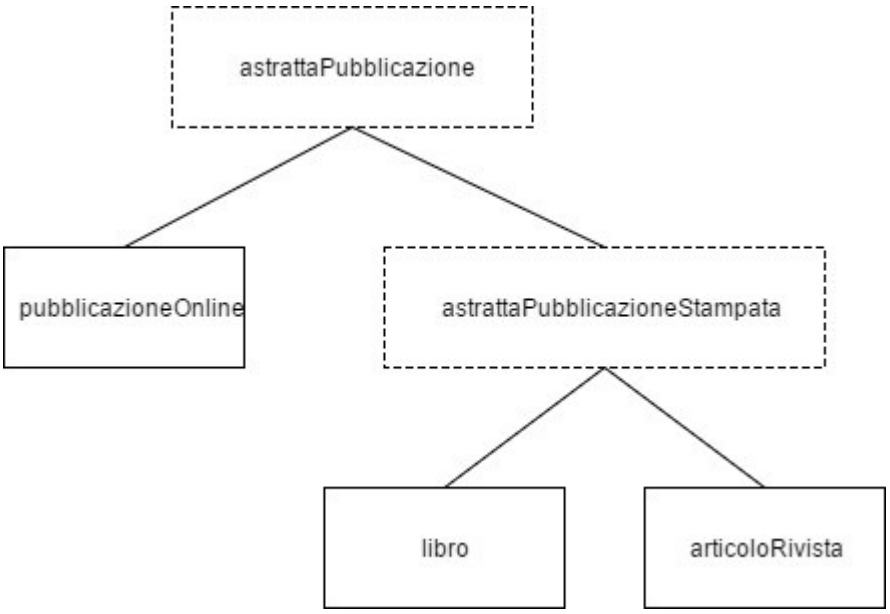
2.1 Gerarchia degli utenti



La gerarchia degli utenti consiste in una classe base utente, dalla quale derivano admin e moderatore. I permessi degli utenti sono gestiti tramite tre funzioni virtuali che restituiscono se un certo utente possa fare o meno un'azione, questo permette di non avere una variabile all'interno degli oggetti che definisce i permessi perché averne una coppia per ogni utente sarebbe stato uno spreco di memoria. Si sarebbe potuto alternativamente utilizzare una

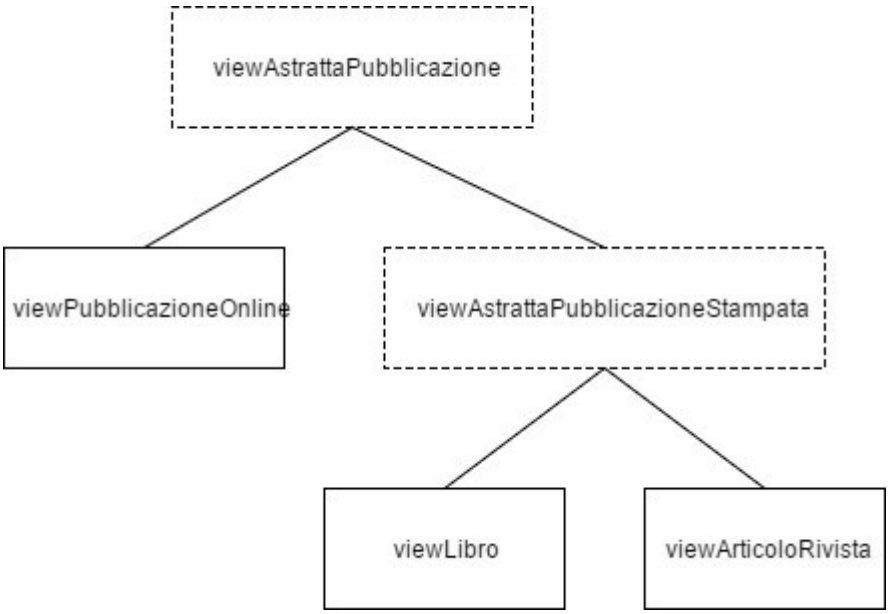
variabile statica per rappresentare i permessi. L'implementazione dei permessi tramite funzioni virtuali permette in caso di estensione della gerarchia la possibilità di creare classi di utenti con permessi dinamici.

2.2 Gerarchia delle pubblicazioni



La gerarchia si basa sulle classi astratte: astrattaPubblicazione e strattaPubblicazioneStampata che definiscono le informazioni basilari di una pubblicazione qualsiasi o di una pubblicazione stampata generica per poi essere completate nelle classi derivate concrete che caratterizzano il tipo di pubblicazione che si sta definendo .La classe astrattaPubblicazione contiene i campi base: titolo ,autore ,doi, materia, descrizione, lingua originale.
La classe pubblicazioneOnLine aggiunge i campi riguardanti il sito ed il link alla pubblicazione e la data in cui è stata pubblicata.
La classe astrattaPubblicazioneStampata aggiunge gli elementi che accomunano le pubblicazioni stampate da una casa editrice, per cui contiene il campo casaEditrice. Libro si caratterizza per avere il campo isbn e l'anno di uscita, mentre per gli articoli di rivista si ha: issn, nome Rivista, numero di uscita e la data di pubblicazione. Le classi possiedono le funzioni virtuali: getCodiceRiferimento() che restituisce il codice di riferimento alla pubblicazione in base al tipo di essa , un metodo di clone(), ed un metodo virtuale per ottenere la data di pubblicazione. Tutte la classi inoltre possiedono la funzione virtuale scriviPubblicazione() che si occupa di scrivere l'oggetto su file xml e la funzione statica importFromXml() che si occupa di leggere da xml e creare l'oggetto.

2.3 Gerarchia vista pubblicazioni



La classe base della gerarchia viewAstrattaPubblicazione, è derivata da QWidget ed ha il compito di creare la GUI adatta ad ogni tipo di pubblicazione, in modo da permettere la visualizzazione di tutti i campi della pubblicazione e di permetterne la modifica o l'inserimento di un nuovo oggetto. La gerarchia inoltre controlla la validità dei dati nel caso l'utente voglia salvare le modifiche o aggiungere il nuovo oggetto.

3 Descrizione dell'uso di codice polimorfo

3.1 Utilizzo del polimorfismo nella gerarchia degli utenti

La gerarchia degli utenti possiede i metodi virtuali isAdmin() , canEdit(), canView() che forniscono le informazioni riguardati i permessi dell'utente, questi metodi vengono usati in mainWidget per costruire l'interfaccia adatta all'utente. È presente il metodo virtuale clone() che crea una copia dell'oggetto.
È stato definito virtuale il distruttore per evitare memory leak nel caso in futuro vengano aggiunti utenti con dei campi dati aggiunti per esempio visto che attualmente le classi derivate da utente fanno solo degli overriding di metodi della classe base.

3.2 Utilizzo del polimorfismo nella gerarchia delle pubblicazioni

Nella gerarchia delle pubblicazioni è presente la funzione virtuale getCodiceRiferimento() che viene usato nel contenitore delle pubblicazioni per controllare che non ci siano pubblicazioni con lo stesso codice di riferimento, è presente come nella gerarchia utenti un metodo di clone ed il distruttore virtuale. È presente una funzione virtuale setDOI che permette in base al tipo dell'oggetto puntato che tipo di controlli effettuare per il settaggio del campo DOI che per alcuni oggetti della classe non può essere nullo perché il codice di riferimento mentre per altri può esserlo.
Sono inoltre presenti funzioni virtuali per scrivere su xml i dati degli oggetti della gerarchia, ottenere il tipo dell'oggetto da scrivere nel file xml e ottenere la data di pubblicazione.

3.3 Utilizzo del polimorfismo nella gerarchia delle viste

In questo caso non è stato ridefinito il distruttore perché essendo una gerarchia che deriva da QWidget viene ereditato già virtuale.
Nella gerarchia sono presenti i metodi caricaCampiDati() che si occupa una volta costruito la dialog per la visualizzazione di inserire i dati dell'oggetto, setEditability(bool) che si occupa di rendere modificabili o meno tutti i campi dati della vista, checkAndSet() che si occupa di controllare la correttezza dei dati inseriti e controlReferenceCode() che estrae il codice di riferimento inserito nella vista addatto.

4 Manuale utente

L'applicazione utilizza due file presenti nella directory dell'eseguibile che vengono utilizzati
per salvare i dati riguardanti gli utenti(utenti.xml) e le pubblicazioni (pubblicazioni.xml).
Nel caso i file sopracitati non siano presenti, viene creato l'utente di default per il login username:admin password:admin, mentre nel caso della mancanza del file pubblicazioni.xml la lista delle pubblicazioni rimane vuota.
La ricerca delle pubblicazioni si effettua selezionando il tipo di ricerca che si vuole tramite i bottoni radio: titolo, autore, codice di riferimento oppure generica che cerca contemporaneamente in tutti e tre i campi elencati precedentemente e digitando il contenuto da cerca all'interno della barra di ricerca.
La lista delle pubblicazioni può essere ordinata in senso crescente e decresente per: titolo, autore, codice di riferimento, anno di pubblicazione con un click

singolo sull'header della colonna per la quale si vuole ordinare gli oggetti, se la colonna è già ordinata viene ordinata nell'ordine inverso.
Per visualizzare tutte le informazioni riguardati la pubblicazione desiderata bisogna effettuare un doppio click su di essa.
E' possibile effettuare il logout per cambiare utente tramite il menu a tendina nella sezione file che permette anche un alternativa
possibilità per la chiusura del programma.

Se l'utente ha i privilegi per modificare la lista e le pubblicazioni può farlo mediante i pulsanti posti nella parte bassa della finestra che permettono: aggiunta, modifica ed eliminazione di pubblicazioni,
alternativamente la modifica è possibile anche dopo aver visualizzato la pubblicazione usando il tasto per attivare
la modifica predisposto nella finestra per la visualizzazione.

Per gli amministratori è presente una sezione extra nel menu a tendina per la gestione degli utenti nel quale è presente la possibilità per l'aggiunta rapida di un utente senza aprire la sezione riguardante gli utenti, oppure si entra nella modalità di gestione degli utenti che permette: aggiunta, modifica ed eliminazione degli utenti.
All'interno della sezione di gestione deli utenti è presente una barra per la ricerca degli utenti

5 Ore utilizzate

- Progettazione: 5h
- Progettazione grafica:3h
- Scrittura codice modello logico:14h
- Scrittura GUI:38h
- Test:2h
- Debug e memory leak: 4h

5 Ambiente di sviluppo

- Sistema operativo: Fedora 25
- Compilatore: gcc 6.3.1 (Red hat 6.3.1-1) e clang 3.9.1
- QT: 5.7.0
- Qmake: 3.0