Relazione sul progetto di Programmazione ad Oggetti a.a 2015/2016

Riccardo Rizzo
1054776

Ambiente di sviluppo

Compilatore: MinGW 32 bit

Versione Qt Creator: 3.4.0

Versione Qt: 5.4.1

Sistema Operativo: Windows 10

Materiale consegnato

La cartella con il materiale consegnato contiene:

1. Tutti i file .h, .cpp e .ui accompagnati da un file .pro necessari per una corretta compilazione del progetto; la cartella localDatabase contiene i file articles.xml e users.xml necessari per memorizzare in modo persistente i dati utilizzati dal database del programma. La cartella html contiene alcuni file .html prelevati dal sito web www.summoner.it e che contengono il codice html di articoli pubblicati nel sito web stesso. Nella cartella icons è presente il set di icone utilizzato per l'interfaccia grafica del programma, mentre in images sono conservate le varie immagini che arricchiscono il testo degli articoli html.

2. Relazione.pdf

Credenziali di accesso suggerite

Lato amministratore	**Username**: admin	**Password**: admin
Lato utente	**Username**: user	**Password**: user

Nel file users.xml sono presenti tutte le credenziali di ogni utente inserito.

0. Compilazione

Per una corretta compilazione del progetto è sufficiente eseguire, nell'ordine, i seguenti comandi:

- qt-532.sh
- qmake
- make

Il progetto viene consegnato con il file .pro già incluso.

1. Introduzione

Il progetto consiste in una riproduzione lato client del sito web www.summoner.it. Il sito in questione è un portale di informazione nel quale vengono pubblicati contenuti inerenti all'universo di videogiochi online, MMORPG ed eSport. Il programma sviluppato mette a disposizione le principali (e più basilari) funzionalità del sito, ossia la visione organizzazata dei contenuti, la possibilità di pubblicare dei commenti specifici per ogni singolo articolo e la gestione base di un sistema di permessi per distinguere le azioni di un normale utente da quelle di un amministratore.

L'organizzazione degli articoli pubblicati nell'applicazione è stata gestita attraverso un'opportuna gerarchia. Esistono infatti tre macro tipologie di articoli: News, Opinioni e Features. Le Opinioni a loro volta si suddivisono in Recensioni e Hands-On.

Il software rende disponibili due tipologie distinte di utenti: utente sottoscrittore (subscriber) e amministratore (moderator). Il login non è obbligatorio per avere accesso alla lettura dei contenuti offerti, ma per poter lasciare un commento, accedere alle funzionalità di eliminazione degli stessi o pubblicare nuovi articoli è necessario effettuare prima l'accesso al servizio. Più nello specifico, un semplice utente sottoscrittore ha la sola capacità di lasciare commenti per un determinato articolo; un amministratore dispone invece dei poteri necessari per eliminare ogni singolo commento e per inserire all'interno del programma nuovi articoli. La gerarchia utile a gestire gli utenti è solo abbozzata per poter distinguire le due tipologie di accesso primarie al servizio, ma è stata strutturata in un'ottica di possibile espansione futura per dotare ogni singolo utente di maggiori funzionalità uniche.

2. Parte Logica

In questa sezione verranno elencate e descritte tutte le classi core sviluppate che fungono da scheletro per la parte logica del progetto, accompagnate da una descrizione delle rispettive funzionalità principali.

2.1 class Container

Come da richiesta, è stata sviluppata una classe contenitore custom denominata Container. La classe è stata utilizzata per immagazzinare tutti gli articoli presenti nel database del programma. Container è stata implementata mediante una lista di nodi (**class Node**) con puntatori al primo e ultimo elemento: Node* first; Node* last; . La classe fornisce metodi di inserimento in testa void insertHead(Article*); , inserimento in coda void insertBack(Article*); e check di controllo bool isEmpty() const; per verificare se la lista è vuota. Inoltre viene fornito un iteratore (**class Iterator**) con definizione di opportuni metodi quali Iterator begin() const;, Iterator end() const; e operatore di accesso a membro Article* operator [](Iterator) const; , utili per scorrere e accedere agli elementi del contenitore.

2.2 class Article

Article è una **classe base polimorfa e astratta** che identifica un oggetto di tipo articolo. **L'oggetto è concretizzato tramite delle sotto categorie**, implementate mediante una gerarchia che si suddivide nelle classi **News**, **Features** e **Opinion**; quest'ultima si ramifica a sua volta in **Review** e **HandsOn**. La gerachia offre un metodo virtuale virtual QString *getCategory*() const; che restituisce la categoria di una specifica classe, in modo tale da evitare svariati *dynamic_cast* altrimenti necessari per i controlli dei tipi dinamici. Ogni oggetto articolo è identificato da una serie di QString: l'ID che identifica il nome del file .html in cui è salvato, il titolo, il genere, la data di pubblicazione e l'autore (ossia l'utente loggato che ha pubblicato tale articolo); è presente inoltre un QVector di puntatori a oggetti Comment che identifica la lista di commenti associati all'articolo. Gli oggetti di tipo Opinion si contraddistinguono per l'aggiunta di un campo dati *summary* che contiene un trafiletto riassuntivo dell'articolo. La classe Review, infine, contiene un intero *score* necessario per conservare il voto della recensione. Non mancano infine varie funzioni di servizio che "espongono" i campi dati privati in sola lettura.

2.3 class Comment

Ogni commento è stato implementato mediante un opportune oggetto di tipo Comment. La classe Comment è costituita da tre campi dati privati di tipo QString che identificano l'utente che ha pubblicato il commento, la data di pubblicazione e il testo contenuto nel commento stesso. I campi dati sono "esposti" in sola lettura mediante opportune funzioni di servizio.

2.4 class User

La classe User è stata implementata per poter gestire gli accessi al programma in modo tale da offrire funzionalità quali l'inserimento e la cancellazione di commenti, e la pubblicazione di nuovi articoli. **La classe User è polimorfa**, e si ramifica in due sottoclassi **derivate pubblicamente** da User: **Subscriber** e **Moderator**. La gerarchia è stata solo abbozzata, ma potrebbe essere ampliata per offrire maggiori funzionalità lato utente. Un oggetto di tipo User è identificato dai seguenti campi dati: un puntatore a un oggetto di tipo **Login**, un puntatore a un oggetto di tipo **Profile** e un boleano che denota se il dato utente ha permessi da amministratore oppure no (bool isMod() const;). La classe Profile contiene un campo dati QString con la biografia dell'utente (non implementato nella GUI ma presente). La classe Login contiene i campi dati QString che identificano l'accesso dell'utente: username, email e password. In Login è stato inoltre ridefinito l'operatore di uguaglianza bool operator==(Login); che si occupa di controllare se due Login sono uguali facendo un controllo diretto sull'username (che identifica univocamente l'utente).

2.5 class Database

Articoli e utenti vengono caricati nella memoria attraverso un oggetto di tipo Database. La classe Database dispone di due campi dati privati: uno di tipo Container (il contenitore custom di puntatori ad articoli) e uno di tipo Qvector<User*> che, come si evince dal tipo, è un contenitore QVector a puntatori di utenti. La classe offre dei metodi di caricamento da file .xml e salvataggio su file .xml. Sono presenti inoltre: un metodo di inserimento di un articolo in testa al Container e due metodi di ricerca che ritornano il primo un contenitore di puntatori ad Article e il secondo un singolo puntatore ad Article, entrambi facendo un match di una QString passata per riferimento costante verso il titolo degli articoli contenuti nel contenitore di Article del Database. Inoltre sono presenti due funzioni di servizio utili per accedere in sola lettura ai contenitori privati del database

I metodi void loadArticles(); void loadUsers(); void saveArticles(); sono stati implementati attraverso l'utilizzo della **libreria QDomDocument**. In void loadArticles(); il file .xml viene letto e caricato in memoria grazie all'ausilio delle funzionalità offerte da QDomDocument, che si occupa di leggere dal file articles.xml le varie informazioni che caratterizzano ogni singolo oggetto articolo. Al contrario, il metodo void saveArticles(); preleva da memoria le informazioni e si occupa di creare un nuovo file .xml contenente tutte

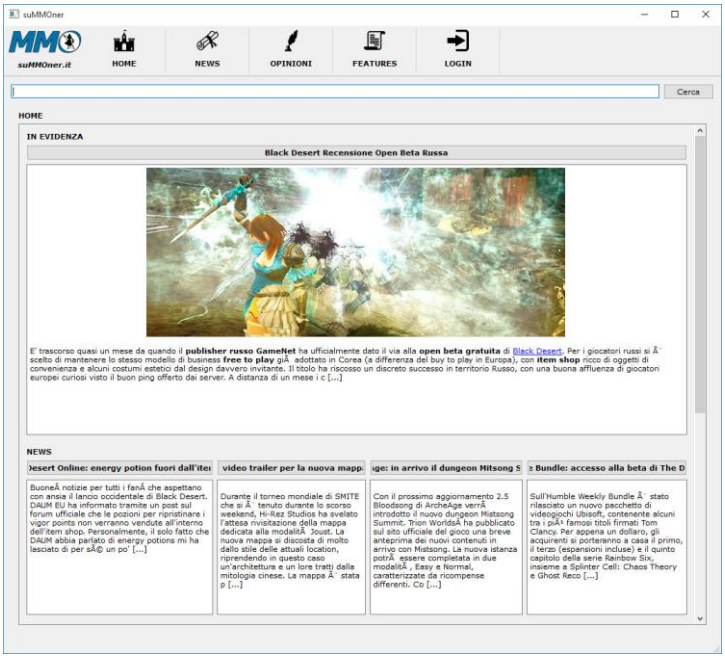
le info di ogni articolo, per poi sovrascrivere il precedente file .xml con quello nuovo. Consapevole del fatto che nel caso di lettura di grosse quantità di dati il caricamento attraverso QDomDocument può risultare in un elevato utilizzo della memoria di sistema, esso è stato preferito a XMLStreamReader per una maggiore semplicità e chiarezza di implementazione ai fini del progetto.

4. Interfaccia Grafica

L'interfaccia grafica è stata realizzata grazie all'ausilio del tool **QtDesigner**. Il comportamento della GUI è stato realizzato collegando le funzionalità realizzate nella parte logica sopra descritta ai vari elementi che costituiscono l'interfaccia grafica. Le tre finestre principali sono:

- MainWindow, che visualizza la schermata principale del programma.
- LoginUI, che rappresenta la schermata di login utile per accedere alle funzionalità lato utente del software
- newArticle, che visualizza la schermata necessaria per inserire un nuovo articolo nel database

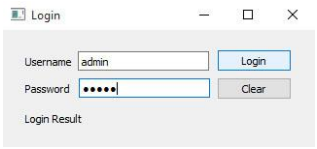
4.1 **MainWindow**



È la finestra principale del programma. Essa è costituita da una toolbar per facilitare la navigazione fra i contenuti offerti e un oggetto QStackedWidget che permette di cambiare la visualizzazione dei contenuti a "schede" in base all'azione cliccata sul suddetto menù. Con la creazione dell'oggetto MainWindow, vengono caricati nel campo dati privato di tipo puntatore a database tutti gli articoli conservati in articles.xml. La schermata Home mostra l'ultima recensione inserita nel database (in altre parole, la più recente). A seguire vengono mostrate le ultime news, gli ultimi articoli di opinione e le ultime features. Alla pressione di un'azione nel menù toolbar si viene portati alla scheda rispettiva a quella categoria di articoli, dove viene mostrato un elenco con tutti gli articoli appartenenti a tale categoria. In queste schermate è presente solo la parte iniziale dell'articolo: per accedere al contenuto completo è sufficiente clickare sul titolo dell'articolo. In questo modo viene aperta una nuova finestra contenente l'articolo completo seguito da tutti i commenti pubblicati in esso. Se si visualizza un'opinione compare anche la sezione Conclusione (data dal campo dati summary di class Opinion) seguita da un voto nel caso di una recensione (data dal suo campo dati score). Per

gestire l'eliminazione dei commenti è stata creata una classe CommentButton (derivato pubblicamente da QPushButton) che serve per passare i puntatori necessari all'eliminazione delle componenti relative a al commento. Alla pressione del bottone di eliminazione, viene cancellato sia il commento dal datbase, che la parte di interfaccia grafica ad esso collegato.

4.2 **LoginUI**



In questa finestra viene presentata la schermata di login necessaria per accedere alle funzionalità utente del programma. LoginUI viene aperta alla pressione dell'azione Login nella toolbar di navigazione. L'oggetto viene costruito passando dei puntatori al database costruito in MainWindow e all'oggetto loginSession contenente la sessione dell'utente che deve loggare. Attraverso la finestra viene effettuato un controllo dei campi dati dell'utente conservati nella classe Login. Se l'utente esiste di presentano due casi:

- se l'utente è un amministratore vengono rese disponibili le seguenti funzionalità: pubblicazione commenti, inserimento nuovo articolo ed eliminazione commenti;
- se l'utente non è un amministratore viene semplicemente resa disponibile la funzionalità di pubblicare commenti all'interno dei singoli articoli.

L'utente viene quindi conservato nell'oggetto puntato da loginSession in modo tale da poter memorizzare le informazioni dell'utente che sta operando all'interno del programma nel corso della sua intera sessione.

4.3 **newArticle**

La finestra utile all'inserimento di un nuovo articolo può essere aperta solamente attraverso la pressione del tasto azione actionNuovoArticolo che compare nella toolbar solamente dopo un accesso corretto da parte di un utente con accessi da amministratore. L'oggetto newArticle viene costruito passando un puntatore al database costruito in MainWindow e all'oggetto loginSession contenente la sessione dell'utente che ha loggato (utile per estrapolare le informazioni relative all'utente che sta per inserire il nuovo articolo). Una volta inseriti correttamente i campi dati all'interno della form e premuto il pulsante di conferma, viene creato un nuovo oggetto articolo memorizzato immediatamente all'interno del suddetto database passato da MainWindows attraverso puntatore. Il salvataggio su file avviene al momento della pressione del pulsante di conferma.