Relazione sul Progetto di Programmazione ad Oggetti

2021/2022

Gabriele Mantoan

1216746

1) Abstract

L'applicazione sviluppata si pone l'obbiettivo di essere una piattaforma per aiutare principalmente gli studenti nell'approccio agli esami universitari (ma anche i professori potrebbero trovarla utile per osservare l'andamento globale delle loro prove). Ispirandosi quindi anche ad altri programmi con la stessa idea (come Uniwhere), prevede una inziale raccolta di feedback contenenti molteplici informazioni riguardo agli esiti degli studenti per poi rielaborarli e dare delle statistiche di facile utilizzo per comprendere le caratteristiche di un particolare tipo di esame.

2) Funzionalità del programma

Il programma permette di leggere i dati da un modello già strutturato in formato .json, oppure di creare un nuovo modello vuoto da cui si può iniziare un nuovo inserimento di dati.
Al fine di testare meglio il progetto è stato fornito un file .json contenente i dati di un modello.

A questo punto vale la pena fare dei chiarimenti su dei termini che compariranno molto spesso:

- Corso: per corso si intende un insegnamento, come ad esempio "Programmazione ad oggetti" oppure "Calcolo numerico"
- Esami/Esiti: per esami o esiti si intende un singolo esito di un esame di un corso, quindi un corso contiene molti esami/esiti.

Dopo aver aperto un file o aver creato un nuovo modello si passera alla "userView", ovvero la schermata principale, dalla quale si possono inserire nuovi corsi, eliminare corsi, modificare corsi, visualizzare i grafici dei vari corsi, aggiungere l'esito di un esame (dopo aver selezionato il corso di appartenenza) e salvare il modello per poterlo riprendere successivamente. Inoltre da questa schermata si può passare alla "esamiView" di uno specifico corso, che permette di vedere tutti gli esiti registrati di quel corso e permette inoltre di modificarli o eliminarli.
Più nello specifico:

UserView:

Aggiunta di un corso: tramite pushButton verrà richiesto il nome del nuovo corso, viene fatto un controllo che il nome non sia già presente e poi crea i bottoni per la selezione, eliminazione e modifica del nuovo corso.

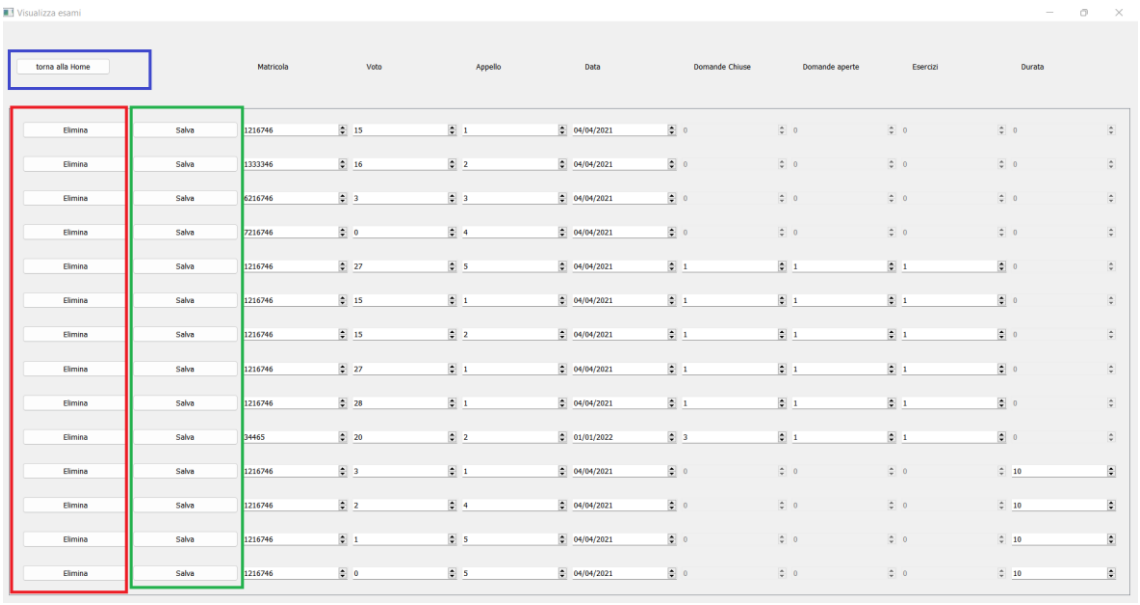
Selezione, modifica ed eliminazione di un corso: la selezione permette di visualizzare in tempo reale i grafici riguardanti a quel esame (e l'eliminazione dei grafici che c'erano precedente se era già stato selezionato un altro corso), la modifica permette di cambiare il nome del corso e l'eliminazione elimina i bottoni riguardanti il corso, oltre ad eliminare i grafici se era stato selezionato

Zona di inserimento esami: qui è possibile inserire un nuovo esito di un esame, ma solo dopo aver selezionato un corso. Per eseguire l'inserimento bisogna compilare i field (matricola, voto, appello e data sono obbligatori) che determinano anche la tipologia di esame che si inserisce (la gerarchia è presentata più avanti), poi premere sul pulsante di aggiunta, che aggiornerà immediatamente i grafici e riporterà i field e dei valori di default.

Salvataggio del modello: se il modello non è nuovo, è possibile salvarlo in due modi: sullo stesso file oppure creando un nuovo file (Salva / Salva come). Se invece il modello è nuovo, entrambe le opzioni si comporteranno allo stesso modo, creando un nuovo file.

Zona grafici: mostra i grafici riguardo il corso selezionato. I grafici possono essere 3 4 o 5, dipendentemente dalle tipologie di esame registrate nel corso: di base sono 3 grafici, se è presente almeno un esame scritto ne viene aggiunto uno e se è presente almeno un esame orale ne viene aggiunto uno.

EsamiView:

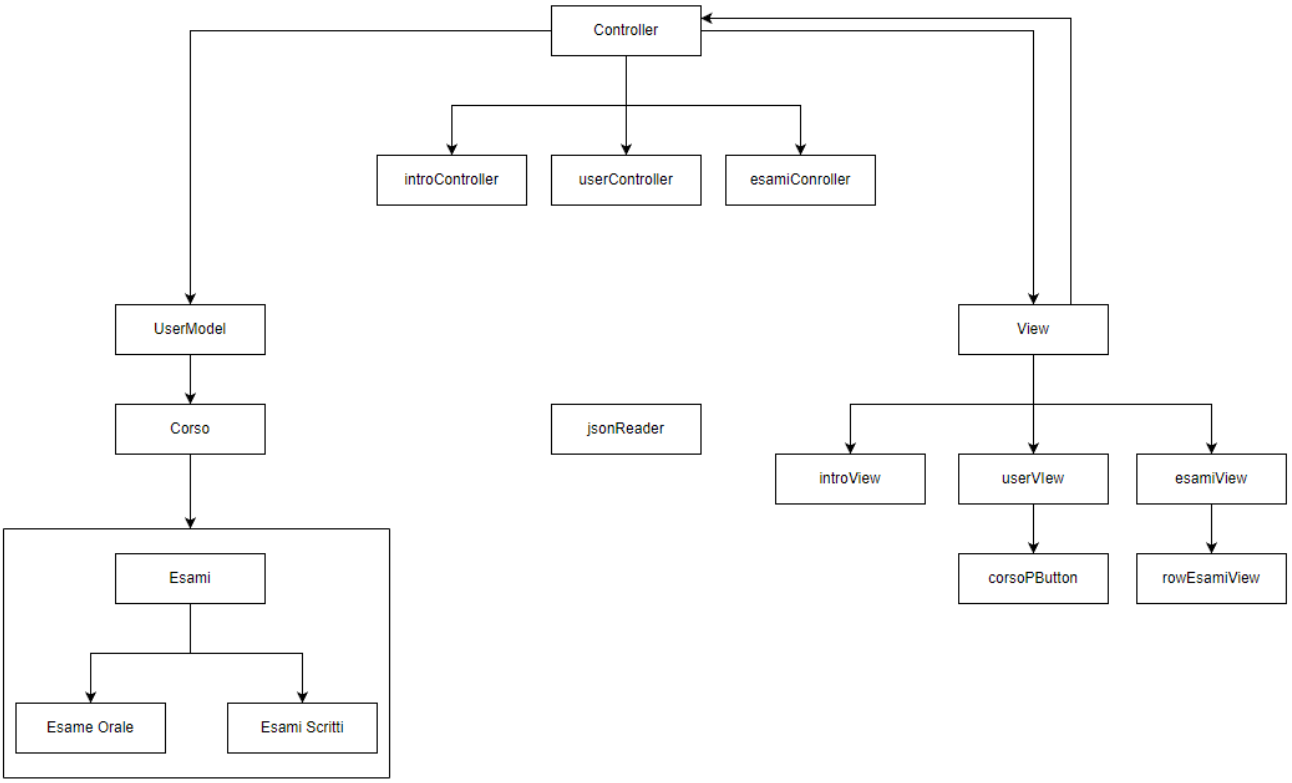


Salva: in questa schermata le caratteristiche di ogni esame vengono presentate in degli appositi field modificabili. Il pulsante salva permette di salvare eventuali modifiche che vengono fatte all'esame
N.B.: certi campi di alcuni esami appariranno oscurati e non modificabili, questo dipende dalla gerarchia di Esame, quindi dopo esser stato inserito, alcuni campi non potranno più essere compilati.

Elimina: permette di eliminare uno specifico esame.

Home: permette di tornare alla schermata precedente (i grafici non verranno aggiornati al fine di permette all'utente di notare la differenza dopo aver cambiato eventuali dati; per aggiornarli basta premere di nuovo sul pulsante del corso)

3) Struttura e gerarchie



Questa è la struttura del programma e delle varie gerarchie, come si può notare ho deciso di utilizzare una il MVC.
Per la vista e per il controller sono state fatte due classi base astratte per poi derivare nelle singole view e nei singoli controller che corrispondono alle 3 schermate dell'applicazione.
Il controller ha dei puntatori verso il modello e verso la vista, che gli permettono di gestire i cambiamenti durante l'esecuzione del programma. Un controller inoltre ha la possibilità di creare un altro controller di diverso tipo per permettere il cambio di schermata.
Come si vede nel grafico, invece, ho scelto di utilizzare un modello unico, contenete un vettore di corsi, i quali a loro volta contengono un vettore di esami.
Gli esiti degli esami sono rappresentati tramite una gerarchia, in cui alla base si trova un esame generico caratterizzato dai seguenti campi: matricola, voto, appello, data. L'esame generico è instanziabile nel caso non esistano dati più specifici oppure non siano recuperabili, ma in realtà la gerarchia deriva in esami scritti (che hanno parametri riguardanti il numero di domande ed esercizi) ed esami orali (che hanno parametri riguardanti la durata di un esame).
Spostandoci dalla parte della View troviamo una gerarchia con base virtuale che si occupa della di mostrare i vari widget a schermo. Sono anche state definite due classi che vengono usate da queste View, corsoPButton, mostra il pulsante di selezione eliminazione e modifica di un certo corso e rowEsamiView, che mostra i pulsanti "salva", "elimina" e tutti i field relativi ad un esame.

Infine è presente una classe statica jsonReader che mette a disposizione i metodi per interagire con i file .json.

4) Esempi di utilizzo di polimorfismo

Alcuni esempi di polimorfismo nel programma sono:

- Il distruttore della gerarchia controller, è stato reso virtuale e permette la distruzione della vista e del modello per tutti i suoi sottotipi
- Il metodo closeEvent di QWidget di cui viene fatto un override nelle view.

5) Modalità di I/O

Il programma usa file .json come formato di input e output, dai quali ricava le informazioni per costruire il modello, cioè sostanzialmente i corsi e i relativi esami. Gli esami vengono già definito come esami semplici, esami scritti o esami orali, in quanto definiti da keywords diverse. Questo è un piccolo esempio:


```
{
    "corsi": [
        {
            "esami": [
                {
                    "appello": 1,
                    "data": "04/04/2021",
                    "matricola": 1216746,
                    "voto": 15
                }
            ],
            "esami orali": [
                {
                    "appello": 1,
                    "data": "04/04/2021",
                    "durata": 10,
                    "matricola": 1216746,
                    "voto": 3
                }
            ],
            "nome": "Programmazione ad oggetti"
        }
    ]
}
```


File che non rispettano il formato .json oppure che non seguono questa sintassi non vengono letti.
Inoltre i file vengono salvati sempre sotto forma di file .json e seguono la sintassi presentata.

6) Monte ore

4 ore per l'ideazione e la progettazione del programma (scelta dell'argomento, decisione delle gerarchie e le principali funzionalità del programma
7 ore per l'apprendimento e lo studio documentazione
6 ore per la creazione del modello
14 ore per la creazione della view
15 ore per la creazione del controller
17 ore per la revisione e testing delle funzionalità

Totale: **63 ore**

Come si può notare il totale del monte ore supera le 50 ore e questo è stato causato principalmente 1 fattore:
Nella prima parte dello sviluppo ho lavorato in modo molto incostante al progetto rendendo molto carente la parte di testing delle funzionalità. Sebbene ne fossi conscio, questo ha portato, verso la conclusione del progetto, una fase di revisione molto intensiva, in quanto le varie componenti del programma non interagivano bene tra loro.

7) Note per la compilazione

Il programma per funzionare necessita dei seguenti pacchetti:
qt5-default
libqt5charts5-dev

Per il resto non necessita di altro, viene fornito il file .pro, può essere compilato con qmake ed eseguito.

Grazie per l'attenzione