RELAZIONE PROGETTO C++

Mattia Bolognini 870401
m.bolognini6@campus.unimib.it

DESCRIZIONE

Il progetto prevede lo sviluppo di una classe template Set, che implementa un container che non può contenere elementi duplicati.

È possibile:
- aggiungere un elemento.
- rimuovere un elemento.
- ottenere un elemento a un certo indice.
- verificare la presenza di un elemento.
- verificare l'uguaglianza di due set.
- stampare un set.
- iterare in sola lettura (iteratore costante) sul set.

Sono definite anche delle funzioni globali che permettono di:
- filtrare un set.
- effettuare la concatenazione di due set.
- effettuare l'intersezione di due set.
- salvare un set di stringhe in un file.

Non sono stati utilizzati costrutti C++11 e oltre, escluso nullptr.

SCELTE IMPLEMENTATIVE

è stato scelto di implementare il container come un array a elementi del tipo generico T.
Ogni set contiene quindi un puntatore a questo array, una capacity (numero totale di celle allocate in memoria) e una size (numero di elementi contenuti).

La quantità di memoria allocata viene modificata dinamicamente: quando il set è pieno, la sua capacità viene automaticamente raddoppiata, mentre quando si svuota, e oltre 3/4 della memoria è libera, la capacità viene dimezzata.
Per modificare la capacità è necessario allocare una nuova zona di memoria, di dimensione desiderata, e procedere con una copia di tutti gli elementi.
La "vecchia" memoria viene poi deallocata.

Non sono state prese altre scelte implementative particolari, in quanto tutto il resto è stato specificato esplicitamente nella traccia.

Espansione della memoria allocata

1) Alloco un set con capacità pari a 6.

2) Inserisco 5 elementi.

3) Inserisco il sesto elemento.

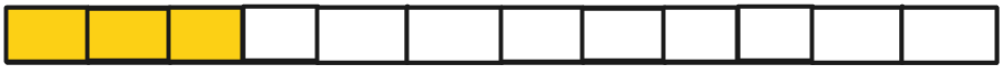
4) Quando viene inserito il settimo elemento, la memoria allocata viene raddoppiata.
Effettuando una nuova allocazione, il set può risiedere in una zona di memoria differente.

5) Dopo aver raddoppiato la quantità di memoria allocata, il settimo elemento può essere inserito.

Riduzione della memoria allocata

1) Si ha un set con una capacità pari a 12 elementi e 7 elementi attualmente inseriti al suo interno.

2) Rimuovo 4 elementi.



3) Rimuovo un altro elemento.

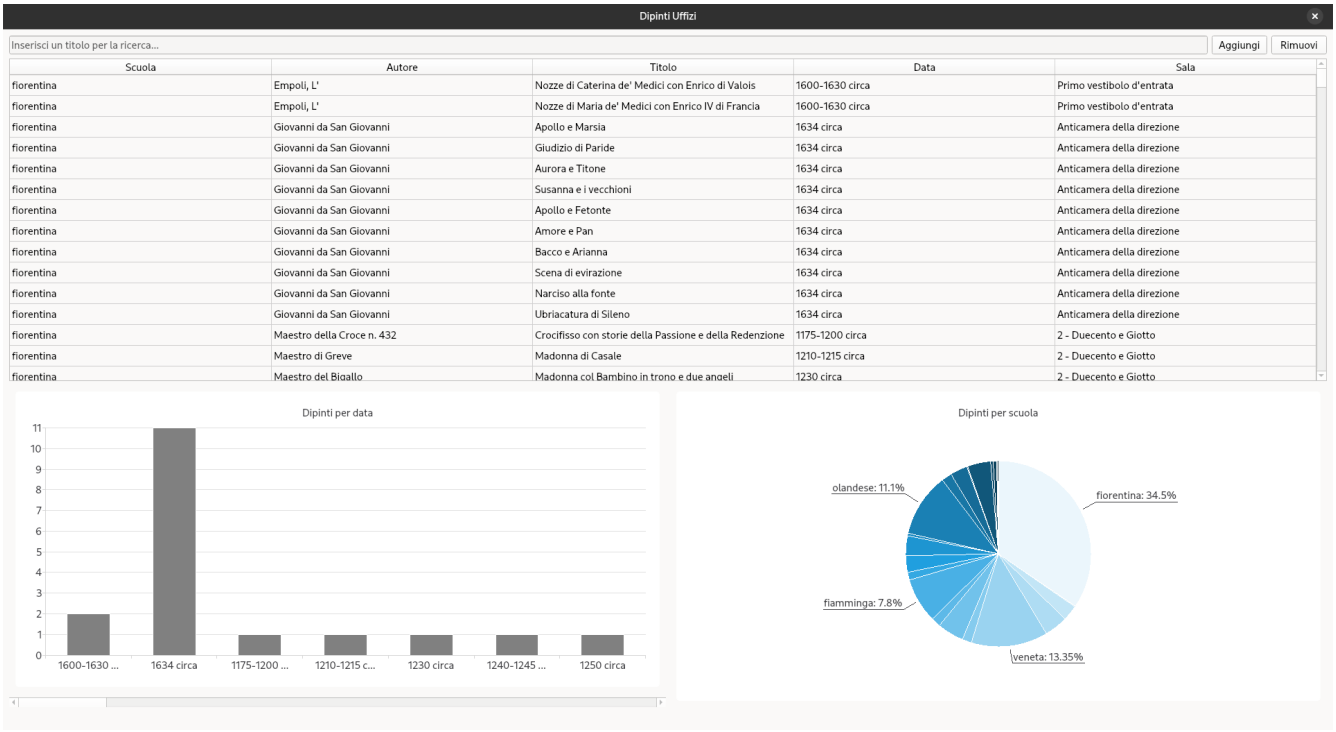
4) Ora la quantità di elementi contenuti è minore di 1/4 della capacità totale del set. Viene quindi dimezzata la capacità, effettuando una nuova allocazione.

RELAZIONE PROGETTO Qt

DESCRIZIONE

Il progetto prevede lo sviluppo di un'applicazione Qt con GUI per la gestione dei dipinti presenti nella Galleria degli Uffizi.
I dipinti sono salvati in un file CSV, "dipinti_uffizi.csv", che viene letto all'apertura dell'applicazione per caricarli al suo interno.
Il contenuto della finestra è adattivo rispetto ad essa, fino a una dimensione minima, oltre il quale non è più possibile rimpicciolirla.
Le principali scelte di design sono state prese per rendere la visualizzazione dei dipinti, la loro aggiunta e la loro rimozione, semplice e intuitiva.
Si è cercato inoltre di evitare di mostrare a schermo o su un'unica finestra troppe informazioni contemporaneamente, in quanto avrebbe reso difficile l'interazione.

INTERFACCIA GRAFICA



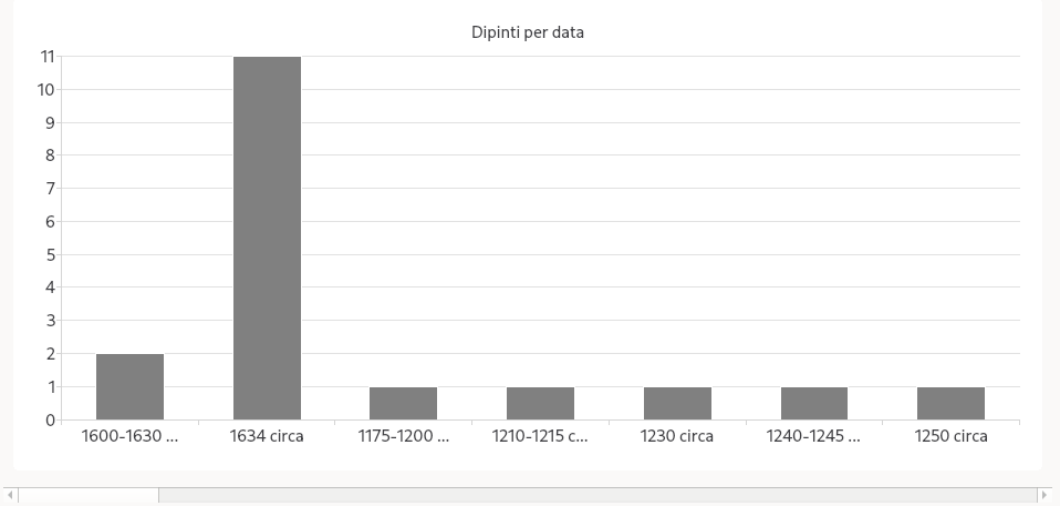
L'interfaccia grafica prevede la presenza di:
- una tabella, per mostrare dettagli relativi a ogni singolo dipinto.
- un grafico a barre, per mostrare il numero di dipinti rispetto al campo Data.
- un grafico a torta, per mostrare la percentuale di dipinti per ogni scuola.
- una barra di ricerca, per filtrare i dipinti mostrati secondo il campo Soggetto/Titolo.
- un pulsante "Aggiungi", per aprire un dialog in cui inserire i dati del nuovo dipinto.
- un pulsante "Rimuovi", per rimuovere tutti i dipinti selezionati sulla tabella.

VISUALIZZAZIONE DEI DIPINTI

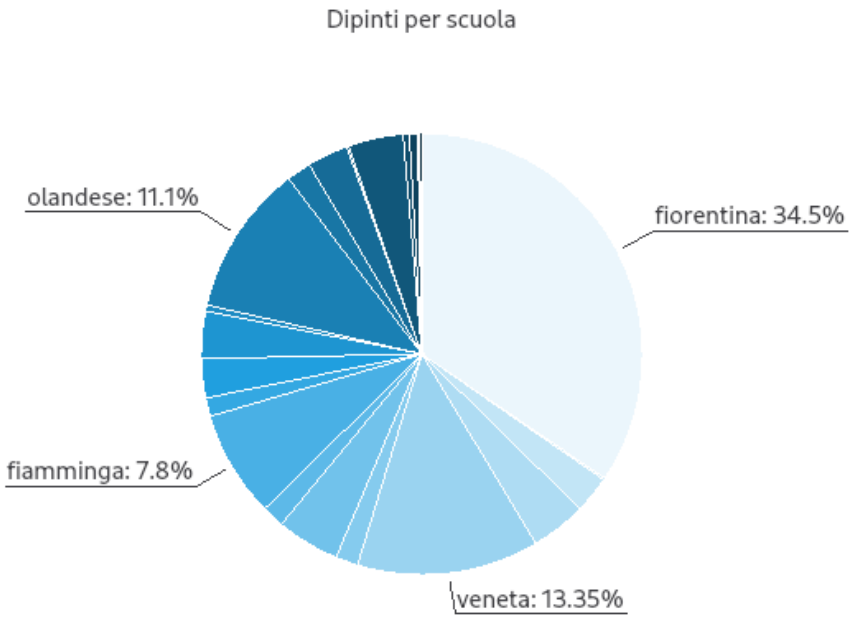
Le informazioni sui dipinti vengono mostrate in tre modi differenti.

Scuola	Autore	Titolo	Data	Sala
fiorentina	Empoli, L'	Nozze di Caterina …	1600-1630 circa	Primo vestibolo …
fiorentina	Empoli, L'	Nozze di Maria de' …	1600-1630 circa	Primo vestibolo …
fiorentina	Giovanni da San …	Apollo e Marsia	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Giudizio di Paride	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Aurora e Titone	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Susanna e i vecchioni	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Apollo e Fetonte	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Amore e Pan	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Bacco e Arianna	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Scena di evirazione	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Narciso alla fonte	1634 circa	Anticamera della …
fiorentina	Giovanni da San …	Ubriacatura di Sileno	1634 circa	Anticamera della …
fiorentina	Maestro della Croc…	Crocifisso con stor…	1175-1200 circa	2 - Duecento e …

Il primo è tramite una tabella: ogni riga mostra i dati relativi a un dipinto, ovvero Scuola, Autore, Titolo, Data e Sala.
I dati presenti nella tabella non possono essere modificati.



Il secondo è il grafico a barre: l’informazione che viene mostrata, in questo caso, è il numero di dipinti per data.
Sull’asse X sono presenti le date, mentre sull’asse Y è presente una scala graduata con tick di distanza 1. Per facilitare la visualizzazione sono inoltre presenti delle linee orizzontali per ogni tick.
La scala viene adattata alla barra con il numero massimo di dipinti, per evitare che il grafico utilizzi una scala fissa che non permette una facile lettura.
Il grafico mostra solo un numero limitato di barre (pari al page step), mentre tutte le altre sono nascoste: per visualizzarle è possibile scorrere una scroll bar orizzontale presente sotto al grafico.



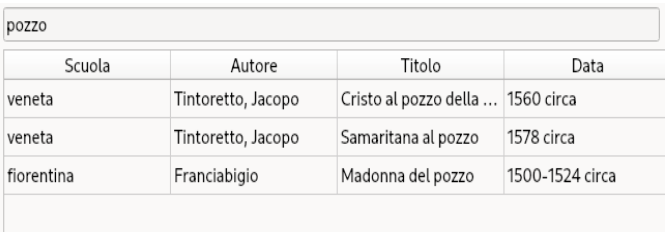
L’ultimo modo in cui vengono mostrati i dipinti è il grafico a torta: ogni fetta rappresenta la percentuale di dipinti che la scuola associata ha prodotto.
A ogni fetta è associata un’etichetta con il nome della scuola e la percentuale rispetto al totale.
Per evitare la presenza di troppe informazioni a schermo, è stato scelto di mostrare solo le etichette delle fette che occupano almeno l’8% del grafico. È comunque possibile visualizzare l’etichetta di una certa fetta (e di conseguenza la relativa percentuale) ed esploderla posizionando il mouse su di essa: tutte le altre fette verranno nascoste, per evidenziare l’informazione che si sta analizzando.

Sia nella tabella che nei grafici, i dipinti mostrati sono quelli che rispettano il criterio di ricerca dettato dalla barra di ricerca.

RICERCA DI DIPINTO/I


Inserisci un titolo per la ricerca...

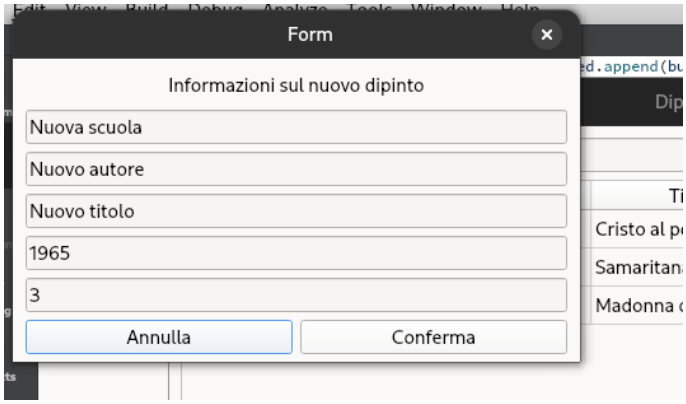
Scuola	Autore	Titolo	Data
fiorentina	Empoli, L'	Nozze di Caterina d…	1600-1630 circa
fiorentina	Empoli, L'	Nozze di Maria de' …	1600-1630 circa
fiorentina	Giovanni da San …	Apollo e Marsia	1634 circa
fiorentina	Giovanni da San …	Giudizio di Paride	1634 circa
fiorentina	Giovanni da San …	Aurora e Titone	1634 circa

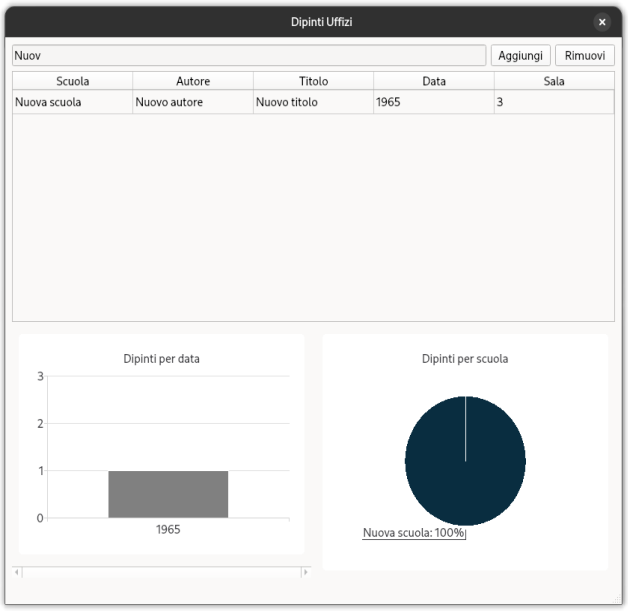

pozzo

Scuola	Autore	Titolo	Data
veneta	Tintoretto, Jacopo	Cristo al pozzo della …	1560 circa
veneta	Tintoretto, Jacopo	Samaritana al pozzo	1578 circa
fiorentina	Franciabigio	Madonna del pozzo	1500-1524 circa

è possibile filtrare i dipinti mostrati utilizzando una barra di ricerca: solo i dipinti il cui titolo contiene ciò che è presente nella barra di ricerca verranno considerati.
è stato scelto di aggiornare i risultati mostrati a schermo a ogni modifica della barra di ricerca, in modo tale da rendere l’operazione più interattiva e “bella da vedere”.

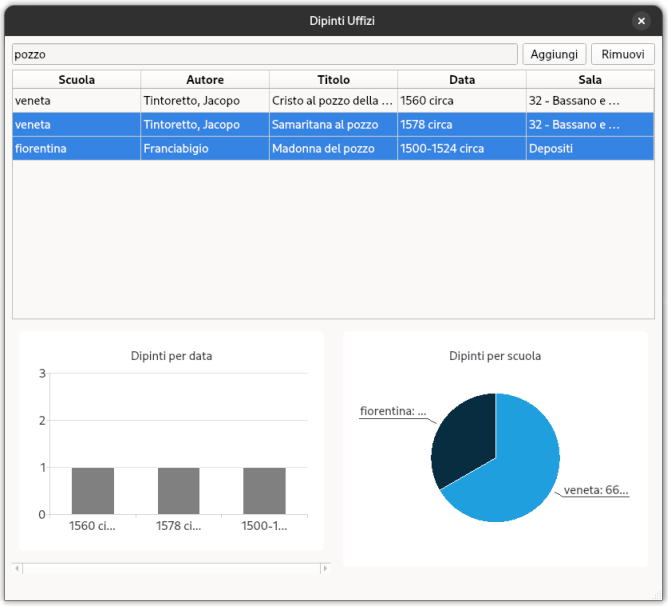
AGGIUNTA DI UN DIPINTO

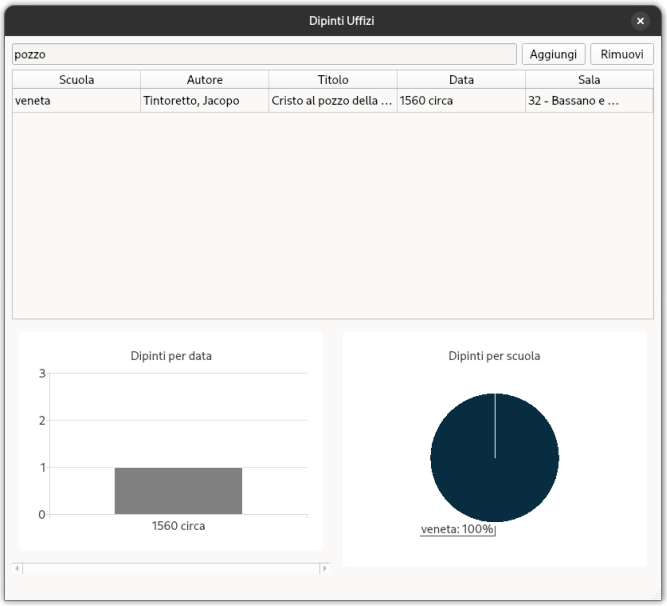




Per aggiungere un nuovo dipinto viene utilizzato il pulsante “Aggiungi”.
Quando premuto, il pulsante crea una nuova finestra Dialog, in cui è possibile inserire I dati relativi al nuovo dipinto: se si sceglie di confermare, il dialog verrà chiuso, il dipinto inserito nella collezione e mostrato poi a schermo; se invece si sceglie di rifiutare, il dialog verrà chiuso e i dati verranno scartati.

RIMOZIONE DI DIPINTO/I





è possibile rimuovere uno o più dipinti contemporaneamente.
Per fare ciò è necessario selezionarli all’interno della tabella e premere il pulsante “Rimuovi”: I dipinti verranno rimossi dalla collezione e, al prossimo aggiornamento dell’UI, non saranno più mostrati.

CODICE

L’applicazione si compone di due parti principali: la main window (form + .cpp + .h) e addDialod (form + .cpp + .h).

MAIN WINDOW

La main window, per il suo corretto funzionamento, utilizza:
- una struct membro “dipinto”, di cui è stato effettuato l’overloading dell’operatore == per permettere il confronto tra dipinti.
- un set “collezione” (implementato con il precedente progetto), utilizzato per contenere oggetti della struct “dipinto”.
- gli slots **on_add_clicked**, **on_remove_clicked**, **on_scroll_bar_scrolled**, **on_slice_hovered**, **updateUI**.
- le funzioni membro private **updatePieChart**, **updateBarChart**, **updateTable**, **updateHorizontalScrollBar**, **readCSV**.

È stato scelto di connettere segnali e slot esplicitamente con la funzione globale connect, evitando quindi il meccanismo di connessione automatica di Qt basato sui nomi dei widget e degli slot.

SLOTS

on_add_clicked

Eseguito quando il pulsante add_button emette il segnale clicked(bool).
Si occupa di allocare e mostrare un nuovo dialog, ottenere le informazioni sul nuovo dipinto da esso e poi deallocarlo.
Chiama poi lo slot updateUI per aggiornare tabella, grafici e scroll bar.

on_remove_clicked

Eseguito quando il pulsante remove_button emette il segnale clicked(bool).
Si occupa di ottenere dalla tabella tutte le righe selezionate ed eliminare i dipinti dalla collezione.
Chiama poi lo slot updateUI per aggiornare tabella, grafici e scroll bar.

on_scroll_bar_scrolled

Eseguito quando la scroll bar associata al bar chart emette il segnale valueChanged.
Aggiorna l’asse X del bar chart: in particolare modifica il suo range di visualizzazione in base alla nuova posizione in cui si trova la scroll bar, ricevuta in input dal segnale emesso.
L’aggiornamento del grafico avviene in automatico alla modifica, non è necessario aggiornare la finestra.

on_slice_hovered

Eseguito quando il cursore del mouse viene posizionato sul pie chart, che di conseguenza emette il segnale hovered.
Riceve in input, dal segnale, la slice il cui stato hovered è stato modificato e un valore booleano che esplicita se la slice è ancora hovered o non lo è più.
Si occupa di gestire la visualizzazione/rimozione delle label delle slices e dell’esplosione della slice hovered.

Le modifiche alle slice aggiornano automaticamente il grafico: non è necessario aggiornare tutta la finestra.

update_UI
Eseguito quando esplicitamente chiamato da altre funzioni membro o quando il testo nella lineEdit (barra di ricerca) viene modificato e di conseguenza viene emesso il segnale textChanged.
Non fa altro che trovare tutti i dipinti della collezione che contengono nel titolo il testo della barra di riicerca (e che quindi possono essere mostrati), per poi chiamare le funnzioni membro di aggiornamento di tabella e grafici.

FUNZIONI MEMBRO PRIVATE
update_pie_chart
update_pie_chart si occupa di deallocare il vecchio pie chart (e oggetti figli) e di crearne uno nuovo aggiornato.
La deallocazione del vecchio grafico viene eseguita con il metodo deleteLater, in modo che venga eliminato all'inizio del prossimo ciclo degli eventi e quindi evitare errori di accesso alla memoria.
Per creare il nuovo pie chart devono essere prima create delle slice (una per ogni scuola), che verrano poi inserite in una bar series.
A ogni slice viene associata una label, con nome e percentuale (troncata alla seconda cifra decimale), che sarà visibile solo se la percentuale è maaggiore dell'8%.

update_bar_chart
update_bar_chart si occupa di deallocare il vecchio bar_chart (e oggetti figli) e di crearne uno nuovo aggiornato.
La deallocazione del vecchio grafico viene eseguita con il metodo deleteLater, in modo che venga eliminato all'inizio del prossimo ciclo degli eventi e quindi evitare errori di accesso alla memoria.
In questo caso è necessario creare anche due assi: un asse di categorie, per contenere le date sull'asse X, e un asse di valori, per rappresentare la scala graduata dell'asse Y.
Viene impostato Il range di visualizzazione dell'asse X in modo tale da mostrare un numero di colonne pari al page step.
L'asse Y viene invece modificato in modo da mostrare una scala graduata con tick ogni unità e il cui valore massimo si adatta alla dimensione della colonna più grande.
Viene poi chiamata updateHorizontalScrollBar per aggiornare la scroll bar associata al grafico.

updateTable
update_table si occupa di aggiornare le entries della tabella, dopo aver deallocato gli oggetti precedentemente contenuti.

updateHorizontalSCrollBar
updateHorizontalScrollBar si occupa di modificare la scroll bar associata al bar chart, in modo tale da rendere il range di valori e il page step adatti al numero di barre presenti nel bar chart.
Il range dipende dal numero di barre totali: se le barre sono meno del page step, allora la scroll bar non potrà scorrere; se invece sono maggiori del page step saranno suddivise in più pagine, e il numero di pagine (che determina il valore massimo della scroll bar) sarà pari al numero di barre diviso il page step.

readCSV

readCSV si occupa di leggere un file CSV contenente i dipinti e di caricarli in un set rappresentante la collezione.
Il CSV utilizza come separatore la virgola ‘,’, mentre per effettuare l’escaping di un campo che contiene il separatore vengono utilizzati i doppi apici.
L’algoritmo di parsing, per distinguere una virgola che si comporta come un separatore da una di cui è stato effettuato l’escaping, verifica il numero di doppi apici contati fino a quel momento: se pari, allora la prossima virgola che si incontrerà sarà un separatore; se dispari, allora la successiva sarà escaped.

AddDialog

L’addDialog contiene solamente una funzione getFields(), che si occupa di restituire una lista contenente i campi letti dalle lineEdit.
Il pulsante “Annulla”, quando emette il segnale clicked, causa l’esecuzione dello slot reject(), che chiude il dialog e ritorna come result della funzione exec il valore 0.
Il pulsante “Conferma”, quando emette il segnale clicked, causa invece l’esecuzione dello slot accept(), che chiude il dialog e ritorna il valore 1.

MEMORY LEAKS

L’applicazione è stata testata con Valgrind e il tool memcheck di Qt per rilevare eventuali memory leaks: sono stati riscontrati solo errori esterni, ovvero i memory leaks sono presenti nelle librerie 3rd-party come quelle Qt, e non sono quindi risolvibili.
L’applicazione dealloca quindi correttamente tutta la memoria che alloca.