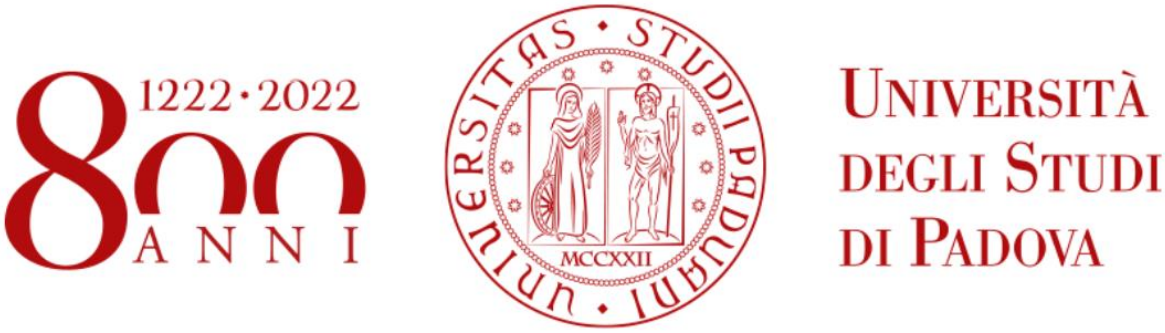


PROGETTO DI PROGRAMMAZIONE AD OGGETTI

ANNO 2021/2022

A cura di Andrea Auletta, 2008458

Introduzione

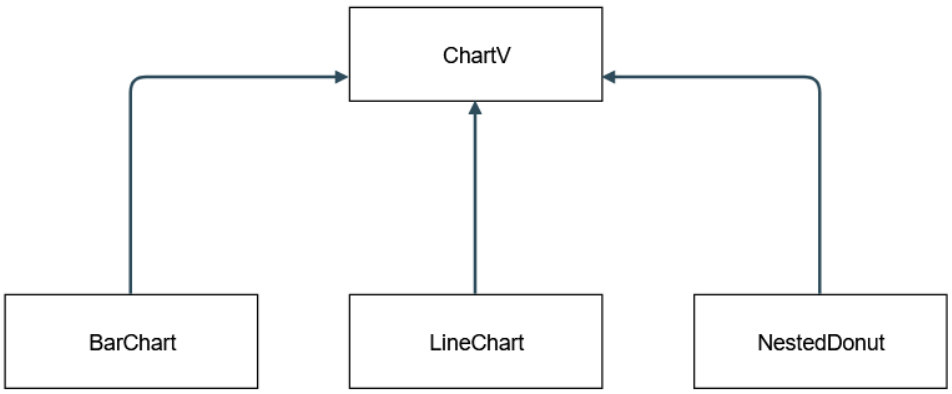
Il progetto consiste in un'applicazione che consente l'inserimento di dati di tipo numerico in una tabella.
Vengono fornite le operazioni di base per la modifica della tabella come l'aggiunta di righe e colonne o la rimozione di esse.

I dati verranno poi rappresentati sottoforma di grafico.
Ci sono tre possibili scelte:

1. Grafico a barre
2. Grafico a linee
3. Nested Donut

Il progetto è stato sviluppato tramite design-pattern "model/view".

Gerarchie



class chartV

E` una classe astratta data la presenza della funzione -> virtual void createC() =0
Sta ad indicare un grafico generico, non può essere istanziato nessun oggetto di questo tipo

Il metodo createC() deve essere quindi implementato dalle classi che ereditano chartV.

class barchart

Eredita chartV e quindi implementa il metodo createC().

Crea il grafico tramite un QVBarModelMapper che prende in automatico i dati dalla tabella anche al momento di nuovi inserimenti e/o modifiche senza necessità di ricostruire il grafico dall`inizio ogni operazione

class linechart

Eredita chartV e quindi implementa il metodo createC().

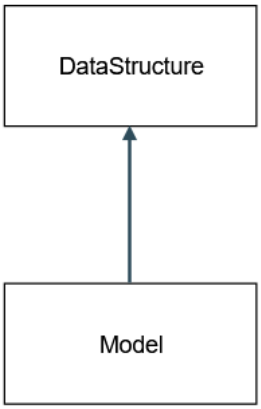
Crea il grafico tramite un vettore di QVXYModelMapper che come scritto sopra permette di aggiornare il grafico in maniera automatica

Viene costruito un vettore perché un singolo QVXYModelMapper corrisponde ad una singola linea del grafico

class nestedDonut

Eredita chartV e quindi implementa il metodo createC().

Utilizza un insieme di QPieSeries per costruire i vari strati del Nested Donut



Class DataStructure e class Model

La classe Model eredita la classe DataStructure dove vengono implementati i metodi per la modifica della tabella e il ritorno determinati valori come ad esempio il valore più grande

Lo scopo è quello di rendere il codice più leggibile tramite le chiamate delle funzioni ereditate da DataStructure

Chiamate polimorfe

La creazione di grafici avviene attraverso la chiamata polimorfa cV-> createC()

La chiamata polimorfa viene usata per semplificare la costruzione dei charts senza il bisogno di specificare il tipo di grafico che si vuole creare

Formato input/output

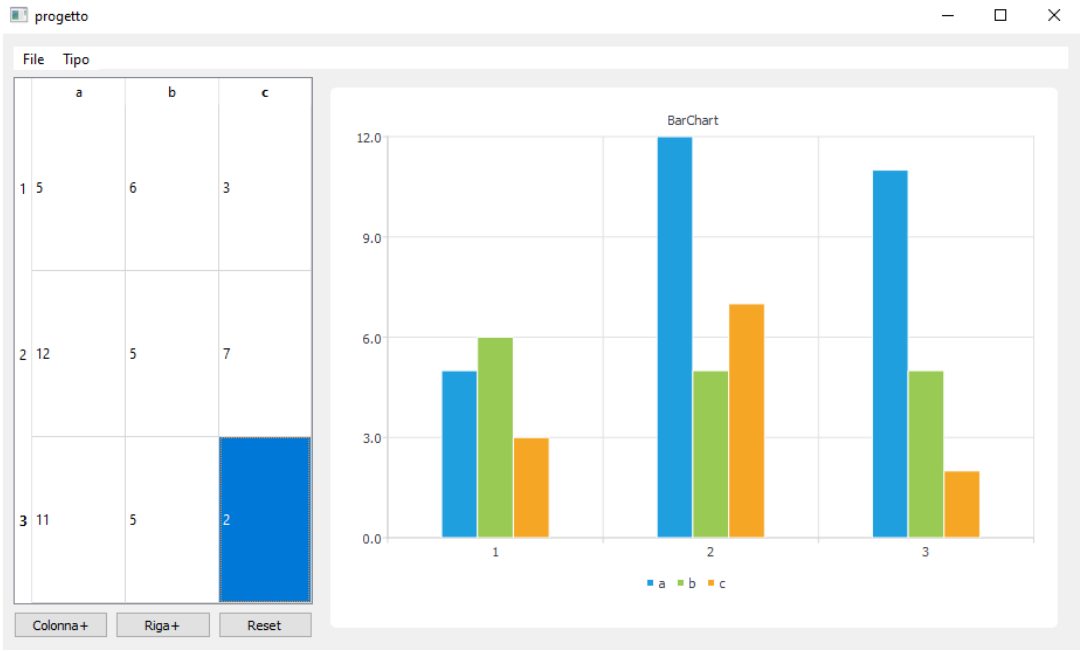
Come formato per il salvataggio e il caricamento di file ho utilizzato JSON
E` stato di facile implementazione: ogni dato presente nella tabella, il numero di colonne e di righe vengono trascritti nel file al momento del salvataggio in modo tale da rendere possibile la ricostruzione della tabella al momento del caricamento del file .json

All'apertura dei file viene eseguito un controllo per ogni dato da cercare per l'inserimento in tabella e per l'accertamento della presenza all'interno del file .json

Nella cartella "Salvataggi" vengono inseriti alcuni file .json per visualizzare il funzionamento dell'applicazione

Manuale utente

Presentazione dell'applicazione

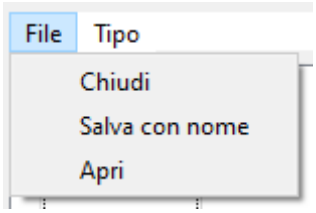


Zona 1: Barra del menu

La barra viene suddivisa in due categorie: File e Tipo

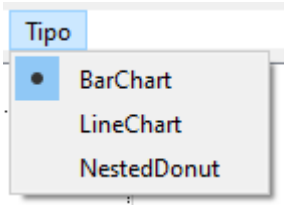
La prima categoria, File, ha 3 possibilità:

1. Chiudi -> chiude l'applicazione
2. Salva con nome -> permette il salvataggio dei dati in un file .json nella posizione che si preferisce
3. Apri -> permette l'apertura e il caricamento di dati da un file .json tramite la ricerca del file stesso
   Dopo il caricamento è necessario selezionare il tipo di grafico che si vuole visualizzare.



La seconda categoria presenta un'opzione per ogni tipo di grafico che si può visualizzare

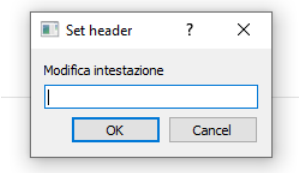
Quindi cliccando su una di queste categorie, automaticamente, il grafico verrà ridisegnato e visualizzato sull'app in base ai dati presenti sulla tabella.



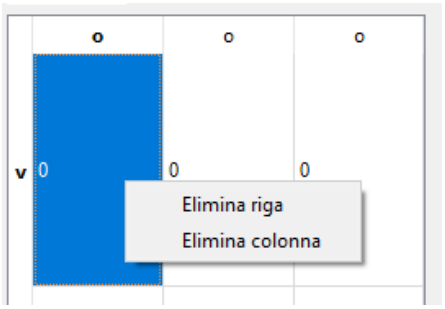
Zona 2: Tabella + Bottoni per la modifica

Il funzionamento della tabella è molto semplice: i numeri possono essere scritti e con un inserimento classico.

Per la modifica degli header, dopo aver cliccato sopra di essi comparirà una finestra di l'inserimento per il settaggio dell'header stesso.
Dopo aver cliccato su "OK" il valore verrà sostituito nella tabella.



Premendo tasto destra su una qualsiasi cella numerica all'interno della tabella comparirà un context menu mediante il quale sarà possibile eliminare la riga o la colonna in cui è situata la cella stessa



Sotto la tabella sono presenti tre bottoni:

1. Colonna+ Aggiunge una colonna alla tabella
2. Riga + Aggiunge una riga alla tabella
3. Reset Resetta tutti i valori presenti nella tabella (imposta a 0 tutti i valori numerici e riporta a 'v' e 'o' i valori degli header)

I tre tipi di grafico vengono presentati in questo modo:

Grafico a barre

Per poter vedere i tre grafici, in questo caso, è necessario modificare il Vertical Header e fare in modo che ci siano valori diversi in modo da non avere grafici con lo stesso nome.

I valori degli assi vengono aggiornati al momento della modifica nella tabella.

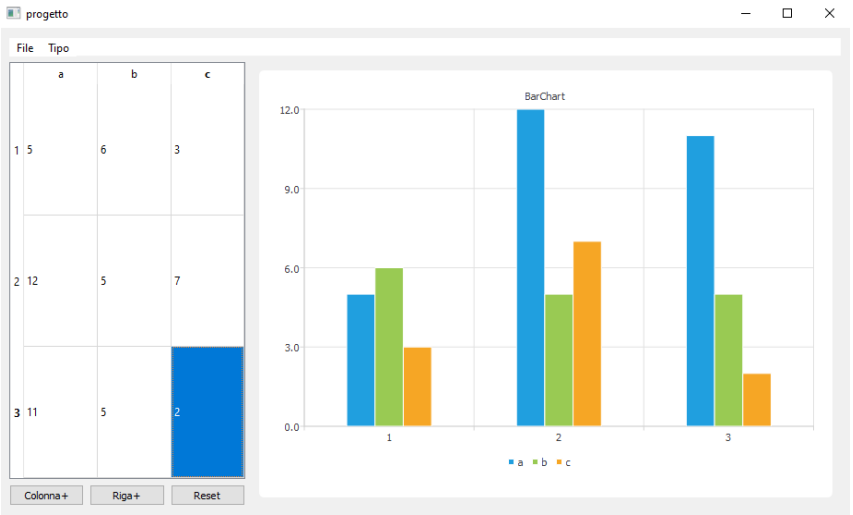


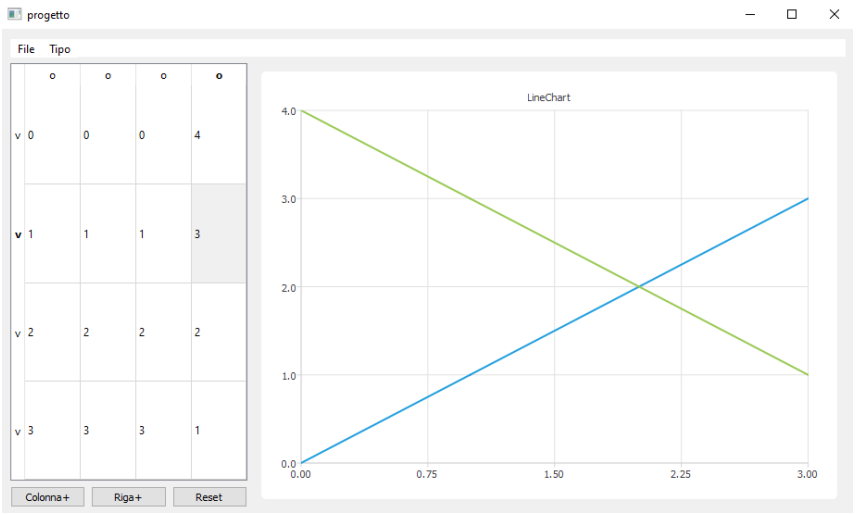
Grafico a linee

Non è necessario modificare i valori degli header.

Una nuova linea viene aggiunta ogni due colonne (per vedere tre linee dovranno essere presenti 6 colonne).

Partendo con la colonna di indice 0, le colonne con indice pari rappresentano i dati dell'asse X mentre le colonne dispari quelli dell'asse Y.

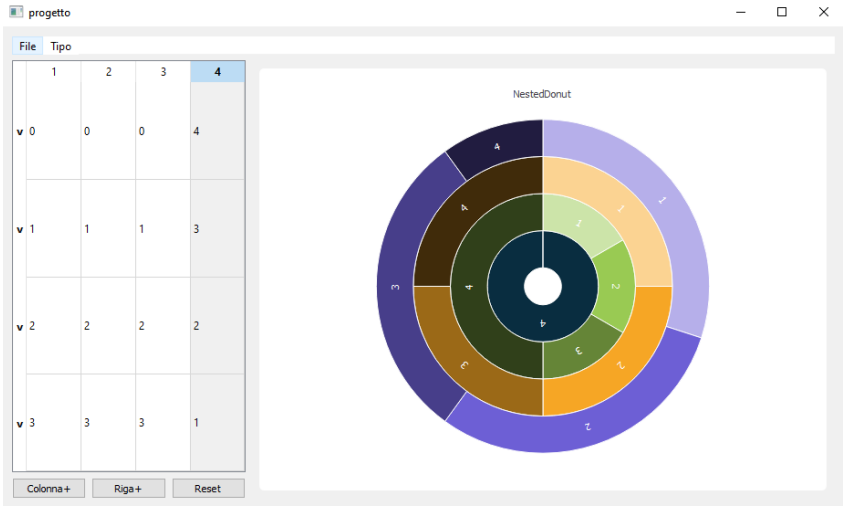
Anche qui gli assi vengono aggiornati al momento dell'inserimento dei nuovi valori.



Nested Donut

Per modificare i label presenti nelle slice del donut è necessario modificare l'Horizzontal Header.

Il grafico viene aggiornato al momento di ogni modifica posta sulla tabella.



Ambiente di sviluppo

Il progetto è stato sviluppato su Windows 10 con Qt 5.9.5

Il programma è stato comunque compilato diverse sulla macchina virtuale messa a disposizione su Moodle.

Compilazione

Per compilare basta accedere alla shell nella cartella del progetto e dare i comandi qmake -> make
Per avviare l'applicazione basta ./progetto

Indicazione ore impiegate per le varie fasi progettuali

1. Analisi preliminare del problema – 2h
2. Apprendimento libreria Qt – 3h
3. Progettazione del modello – 22h
4. Progettazione della GUI – 18h
5. Debugging – 4h
6. Testing – 2h
7. Relazione - 2h

Totale: 53h