UNIVERSITÀ
DEGLI STUDI
DI PADOVA

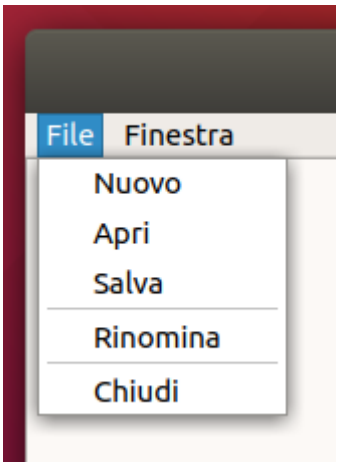
PROGETTO DI PROGRAMMAZIONE AD OGGETTI

ANALISI CHIMICA
DI CAMPIONI E ELEMENTI

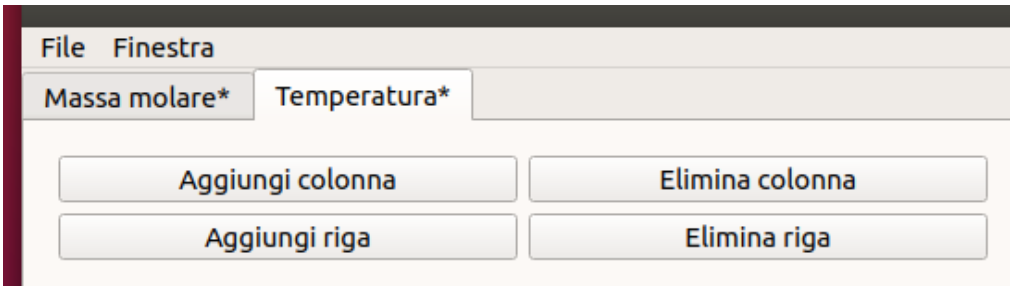
ANNO ACCADEMICO 2021/2022

RELAZIONE DI CAVALIERE ERICA - 2013450

1. INTRODUZIONE



Il programma “Grafici per Analisi Chimica” è stato pensato per chi lavora nei laboratori di analisi, dove si analizzano campioni per la produzione di materiali chimici, tessili, prodotti alimentari (per la ricerca di allergeni o per riportare la percentuale di presenza di un determinato elemento) o anche in campo medico, per analisi specifiche come le analisi del sangue.
L’applicazione si presenterà come una finestra bianca con disponibile una barra dei menu nel quale sarà possibile scegliere se creare, aprire, salvare i grafici presenti nel programma o rinominarli.
Verrà resa disponibile una TabBar, attraverso la quale sarà possibile cambiare la visuale tra i grafici aperti o creati dall’utente.
Ogni finestra avrà a destra il grafico, che potrà essere a torta, a linea o a punti, mentre a sinistra ci sarà una struttura tabellare che potrà essere modificabile dall’utente, potendo così aggiornare il grafico accanto.



2. PREMESSA

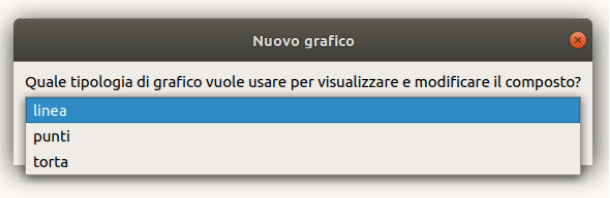
Insieme al programma è stato dato un file “elementi.json” dove sono stati salvati alcuni elementi della Tavola Pitagorica e sono utilizzabili per provare a creare nuovi grafici.
Per poter essere letto dal programma, bisogna posizionare il file nella cartella Home dell’utente che esegue il programma e sarà così possibile leggere o modificare il file.
Se questo file non è presente potrebbero comparire degli avvisi di errore, ma sarà lo stesso possibile creare i grafici desiderati.
Se non si dispone del file citato sopra, basta creare un nuovo elemento attraverso il percorso “File → Nuovo → Elemento” e il programma si occuperà di aggiungere in automatico il file “elementi.json”.
Per questione di tempo non è stato possibile inserire alcune funzioni, ma che potrebbero essere poi implementati in futuro e, avendo creato l’applicazione per il progetto di Programmazione ad oggetti, è stata fatta una ricerca dei termini tecnici da un punto di vista chimico per creare il modello e le funzioni principali, ma potrebbero non essere precise, quindi il programma non è da considerarsi definitivo per un uso reale.
E’ stato fatto principale affidamento al libro “Studiare Chimica” di Paolo Pistarà, edizione del 2019, di seguito vengono date le definizioni di alcuni termini utilizzati all’interno del programma:

- Campione: una materia soggetta ad esperimenti.
- Elemento: una sostanza costituita da un solo atomo.
- Composto: una sostanza costituita da atomi di diversi elementi.
- Miscela: un campione di materia formato da uno o più composti
- Mole: l’unità di misura definita dal Sistema Internazionale per la quantità di una sostanza che definisce una quantità precisa di particelle elementari (cioè atomi).

3. FUNZIONALITA’

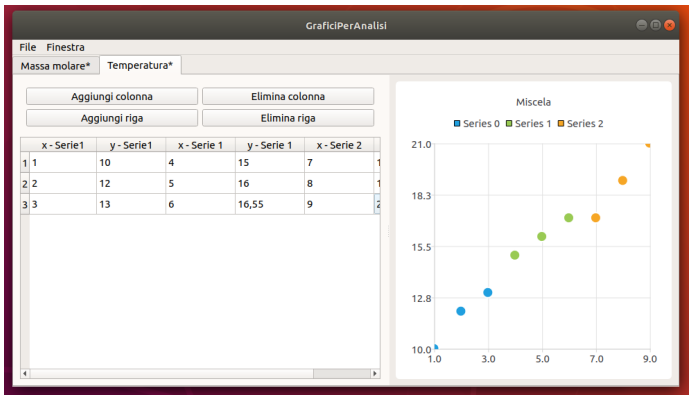
All’apertura del programma si presenterà una finestra bianca con presente solo una barra dei menu con i pulsanti File e Finestra.
Attraverso la voce File sarà possibile:

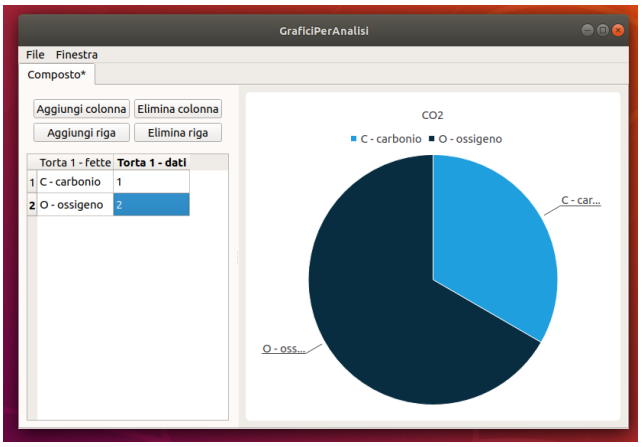
- Creare un nuovo Elemento (seguendo il percorso File → Nuovo → Elemento). Gli elementi creati saranno poi resi disponibili per creare un composto.
- Creare il grafico di un composto o di una miscela. Per creare una miscela, verrà chiesto di selezionare uno ad uno i composti salvati in formato json. Per uscire dalla finestra di selezione dei file, basterà cliccare annulla.
- Creare i grafici per visualizzare la massa molare di una miscela, le temperature dei composti che formano una miscela o visualizzare la Molarità (cioè il rapporto delle moli di un campione con il volume della miscela). Per poter calcolare la molarità o la massa molare bisogna che la miscela sia una soluzione, altrimenti non sarà possibile nemmeno creare i grafici appena nominati.
- Selezionare il modello con il quale si desidera visualizzare i grafici, sarà l’ultima cosa che verrà chiesta alla fine delle creazione di un nuovo grafico.

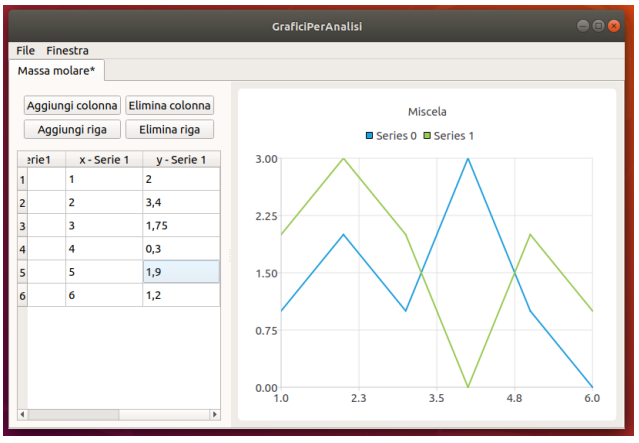


- Aprire un file.
- Salvare i file aperti e visibili attraverso la TabBar.
- Rinominare il grafico attualmente visualizzato nella TabBar
- Chiudere il programma.

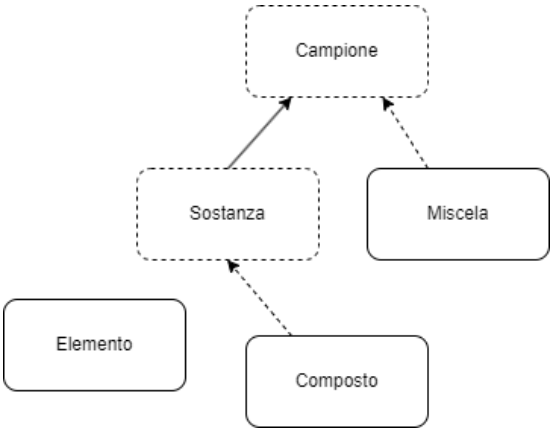
Nella voce Finestra sarà invece possibile modificare la posizione della leggenda e decidere se visualizzare dei grafici animati.
Infine, come già accennato precedentemente, sopra ad ogni tabella ci saranno 4 pulsanti che permetteranno di aggiungere o eliminare le righe e le colonne della tabella attualmente visibile; non sarà data la possibilità all’utente di scegliere in quale posizione verranno aggiunte e quando vengono cliccati i rispettivi pulsanti “Elimina” sarà eliminata la prima riga dal basso o la prima coppia di colonne a destra.
Una coppia di colonne corrisponde a una serie di dati dove la prima colonna indica la riga delle x, mentre la seconda la riga delle y (secondo il grafico a punti e a linea), oppure a una torta, dove le prima colonna rappresenta i nomi delle fette e la seconda colonna i dati.







4. GERARCHIA DEI TIPI



Nel grafico qui a fianco viene indicata la gerarchia di tipi utilizzata nel modello, dove le classi tratteggiate indicano le classi astratte, mentre le frecce tratteggiate indicano una derivazione virtuale.
Alla base della gerarchia si trova la classe **Campione** che implementa i metodi get() e set() che permettono di interagire con i campi dati comuni per tutti gli oggetti derivati.
La classe **Sostanza**, anch'essa astratta, rappresenta una materia costituita da uno o più specie chimiche (ovvero da uno o più elementi), include il campo dati nMoli e massaMolare, entrambe di tipo double, ma la massa molare viene definita come il rapporto tra la massa del campione e il numero di moli (nMoli).
Un **Elemento,** in chimica, viene indicato come una sostanza formata da una sola tipologia di atomi, ma in questa gerarchia è stato scelto di tenerla come classe separata perchè contiene le caratteristiche di un atomo, che sono diverse da quelle indicate nella classe Campione.
Dell'atomo, non è possibile raccogliere la temperatura o il volume, mentre la massa viene indicata come massa atomica e infine, l'atomo, viene rappresentato dal numero atomico, che permette di indicare dove si trova l'atomo nella Tavola Periodica degli elementi.
La classe **Composto** viene utilizzata come un "vettore" di Elementi. Un composto formato da un solo oggetto nell'array, rappresenta l'Elemento, secondo il termine tecnico descritto poco fa.
Il Composto include il campo dati formulaChimica di tipo string e sono state inserite le seguente funzioni:

- massaElemento(const Elemento& e) consiste nel calcolare la massa rappresentata da un oggetto Elemento in quel composto, cioè non la massa atomica, si va a controllare la massa formata da un tipo di particelle nel composto;
- percentualeMassa(const Elemento& e) calcola la percentuale della massa rappresentata da un oggetto Elemento nel composto;

La classe **Miscela** invece viene utilizzato come un "array" di Composti e può essere una soluzione, ovvero una miscela formata da particelle che è difficile distinguere tra loro a occhio nudo, un esempio è l'acqua con il vino, mescolate insieme non è possibile capire quali sono le molecole dell'acqua e quali sono le molecole del vino (in questo caso il vino e l'acqua vengono chiamati anche soluti), mentre un esempio di miscela che non è una soluzione, è l'acqua mischiata insieme all'olio, dove invece è possibile distinguere ad occhio nudo i due composti.
Solo quando una miscela è una soluzione è possibile applicare alcune formule, implementate nelle seguenti funzioni:

- massaSoluto(const Composto&) calcola la massa rappresentata da un oggetto Composto nella Soluzione
- volumeSoluto(const Composto&) calcola il volume rappresentato da un oggetto Composto nella soluzione
- moliSoluto(const Composto&) calcola il numero di moli rappresentati da un oggetto Composto nella soluzione
- percentualeMassa(const Composto&) calcola la massa in percentuale rappresentata da un oggetto Composto nella soluzione
- percentualeVolume(const Composto&) calcola il volume in percentuale rappresentato da un oggetto Composto nella soluzione
- massaSuVolume(const Composto&) calcola il rapporto tra la massa di un Composto e il volume della Miscela
- molarita(const Composto&) calcola la percentuale del rapporto tra le moli di un Composto e il volume della Miscela (molarità).

5. GRAPHICAL USER INTERFACE

La GUI (Graphical User Interface) viene gestita dalla classe **MainWindow**, dove sono stati definite delle funzioni private che si occupano di creare i grafici e le tabelle che poi saranno visibili all'utente e vengono anche gestite le funzioni delle opzioni presenti nel menu a tendina del pulsante "Finestra", la funzione Rinomina (File → Rinomina) e i tasti "Aggiungi colonna", "Elimina colonna", "Aggiungi riga", "Elimina riga".
Un oggetto MainWindow è un oggetto QMainWindow e sono stati creati diverse variabili per gestirlo, ma il campo dati più importante è window, ovvero un QVector<DatiGrafico> che contiene in ordine i dati dei grafici resi visibili e disponibili tramite il QTabWidget.
La classe **DatiGrafico** ha lo scopo di gestire i dati di un grafico, infatti al suo interno vengono definiti diversi tipi:

- DatiPunto rappresenta il punto di un grafico a linea o a punti, oppure una fetta del grafico a torta, infatti è formato dalla coppia di tipi QPointF per gestire la posizione sugli assi x e y, mentre QString indica il nome del punto o della fetta; in MainWindow per creare una fetta del grafico a torta, vengono utilizzati i campi QString e il campo y del QPointF.
- DatiLinea è una QList di DatiPunto, rappresenta una serie di dati che insieme formano una linea del grafico o una serie di punti o una torta.
- DatiTabella è una QList di DatiLinea, permette di rappresentare il grafico vero e proprio, tutte le linee, tutte le torte e tutte le serie di punti che sono stati definiti e che rappresentano il grafico intero. Il nome DatiTabella è stato scelto perchè i dati effettivi non sono visibili nel grafico stesso, ma lo sono attraverso la QTable definita nel MainWindow e tramite la tabella grafica è possibile modificare i dati definiti in DatiTabella.

In DatiGrafico oltre ai dati visibili nella tabella, contiene anche l'enum TipoGrafico, che permette di definire se il grafico è a torta, a linea o a punti, e infine i dati di tipo QString titolo e intestazione che rappresentano rispettivamente il titolo del grafico e il nome nel TabBar.
Tornando a parlare della classe MainWindow, è possibile implementare la view anche in altri progetti, infatti, oltre a rendere disponibile la funzione aggiungiGrafico() che permette di aggiungere una nuova scheda al QTabWidget, sono stati definiti tre signals che permettono di gestire i pulsanti "Nuovo", "Apri" e "Salva" presenti nel menu a tendina "File".
Infatti, nella classe Controller vengono utilizzati questi segnali per permettere il salvataggio e la lettura dei dati in JSON.

6. POLIMORFISMO

Nella classe Campione sono state dichiarate e resi virtuali i seguenti metodi:

- virtual void setNome(std::string);
- virtual void setMateria(Stato);
- virtual void setMassa(double);
- virtual void setVolume(double);
- virtual void setTemperatura(double).

Queste funzioni vengono utilizzate in modo comune da tutte le classi derivate, eccetto la funzione virtual void setMassa(Stato) che è stata definita nella classe Sostanza in modo che venga fatto un controllo alla variabile double massaMolare e, di conseguenza, anche nella classe derivata Composto.
Sempre nella classe Campione, sono state rese virtuali pure le seguenti due funzioni che sono state poi implementate nelle classi concrete in modo da evitare *memory leak*:

- virtual Campione* clone() const;
- virtual ~Campione();

Ipotizzando che in un futuro si possano scoprire nuove tipologie di campioni, miscele o composti, è stato scelto di rendere il modello estendibile, per questo motivo è stato scelto di rendere la derivazione virtuale in Composto e in Miscela

7. FUNZIONALITA' DI INPUT/OUTPUT

Per la gestione e il salvataggio dei dati, viene utilizzato JSON.
La classe **Controller** si occuperà a gestire la lettura e scrittura dei file e ci saranno 2 tipi di file distinti su cui il programma lavorerà.
Il primo file consiste nel registrare tutti gli elementi salvati dall'utente in elementi.json, già nominato in precedenza, ogni Elemento sarà salvato come un oggetto identificato da una chiave che, per convenzione, sarà il numero atomico, e per tenere traccia degli elementi salvati verrà creato un array "chiavi": il programma controllerà se il file rispetterà la seguente struttura {"chiavi":[...]}.
La seconda tipologia di file gestita dal programma in JSON consiste nella rappresentazione di tutti i grafici creati dall'utente, il file sarà un oggetto che conterrà tutti i dati della miscela o del composto e avrà poi al suo interno un'altro oggetto definito dalla parola chiave "grafico", che conterrà invece tutti i punti dei grafici definiti; l'unico controllo svolto dal programma sarà se il programma rispetta la seguente struttura {"file":..} dove file va ad indicare la tipologia di oggetto salvata, cioè "composto", "miscela" o uno dei grafici per visualizzare i dati della massa molare, la temperatura o la molarità.
Se vengono modificati i file in modo da evitare dei controlli o in modo che non vengano rispettati determinati tipi di parametri, è possibile andare in contro a dei *segmentation fault* con conseguente crash del programma.

8. ISTRUZIONI DI COMPILAZIONE ED ESECUZIONE

Insieme al progetto è stato fornito il file GraficiPerAnalisi.pro, deve essere usato questo file per poter eseguire il codice e bisogna anche aver installato il pacchetto: **qt5-defaultlibpq5charts5-dev**
Per compilare il programma bisognerà eseguire le seguenti istruzioni:
qmake -o GraficiPerAnalisi.pro Makefile
make -o GraficiPerAnalisi.pro
Infine, per eseguire il codice, basta eseguire la seguente istruzione:
./GraficiPerAnalisi

9. AMBIENTE DI SVILUPPO

Il progetto è stato sviluppato su Windows 10 con compilatore MinGW 5.3.0 e libreria Qt 5.9.5.
Inoltre è stato testato con la macchina virtuale fornita dal docente che offre il sistema operativo Ubuntu 18.04.3 LTS, compilatore GCC 64-bit e libreria Qt 5.9.5

10. ORE DI LAVORO

- Creazione del modello: 10 ore
- Creazione della GUI: 16 ore
- Creazione del controller: 13 ore
- Implementazione JSON: 6 ore
- Debug e testing: 6 ore
- Ore totali: 51 ore

Il motivo del perchè è stata sforata la soglia oraria stabilita è perchè sono stati riscontrati diversi problemi durante lo sviluppo del programma stesso, andando a gestire diversi segmentation fault,, in particolare sulla lettura e sulla gestione dei dati, risolti andando a cercare gli errori attraverso il debug.
Nel conteggio delle ore non sono state prese in considerazione le ore spese nello studio della documentazione di Qt e la ricerca generale per la progettazione del modello.