GraphViewer

Progetto sviluppato da Massimo Chioru 2013449 e Elena Fabris 2008072

Relazione Progetto di Programmazione ad Oggetti

Massimo Chioru 2013449
a.a. 2021/2022

Indice

1 - Introduzione **2**
1.a - Funzionalità 2
1.b - Screenshot rappresentativo del software 2

2 - Descrizione delle gerarchie dei tipi **3**
2.a - Il Modello 3
2.b - BaseChart 3
2.c - View 4
2.d - Controller 4
2.e - Doc e About us 4

3 - Descrizione dell'uso delle chiamate polimorfe **4**
3.a - BaseChart 4

4 - Descrizione dei formati di input e output **5**
4.a - JSON (JavaScript Object Notation) 5

5 - Manuale utente della GUI **6**
5.a - Descrizione 6

6 - Istruzioni di compilazione ed esecuzione **6**
6.a - Prerequisiti 6
6.b - Istruzioni 6

7 - Descrizione delle ore richieste per task **7**

8 - Suddivisione del lavoro progettuale **8**

1 - Introduzione

1.a - Funzionalità

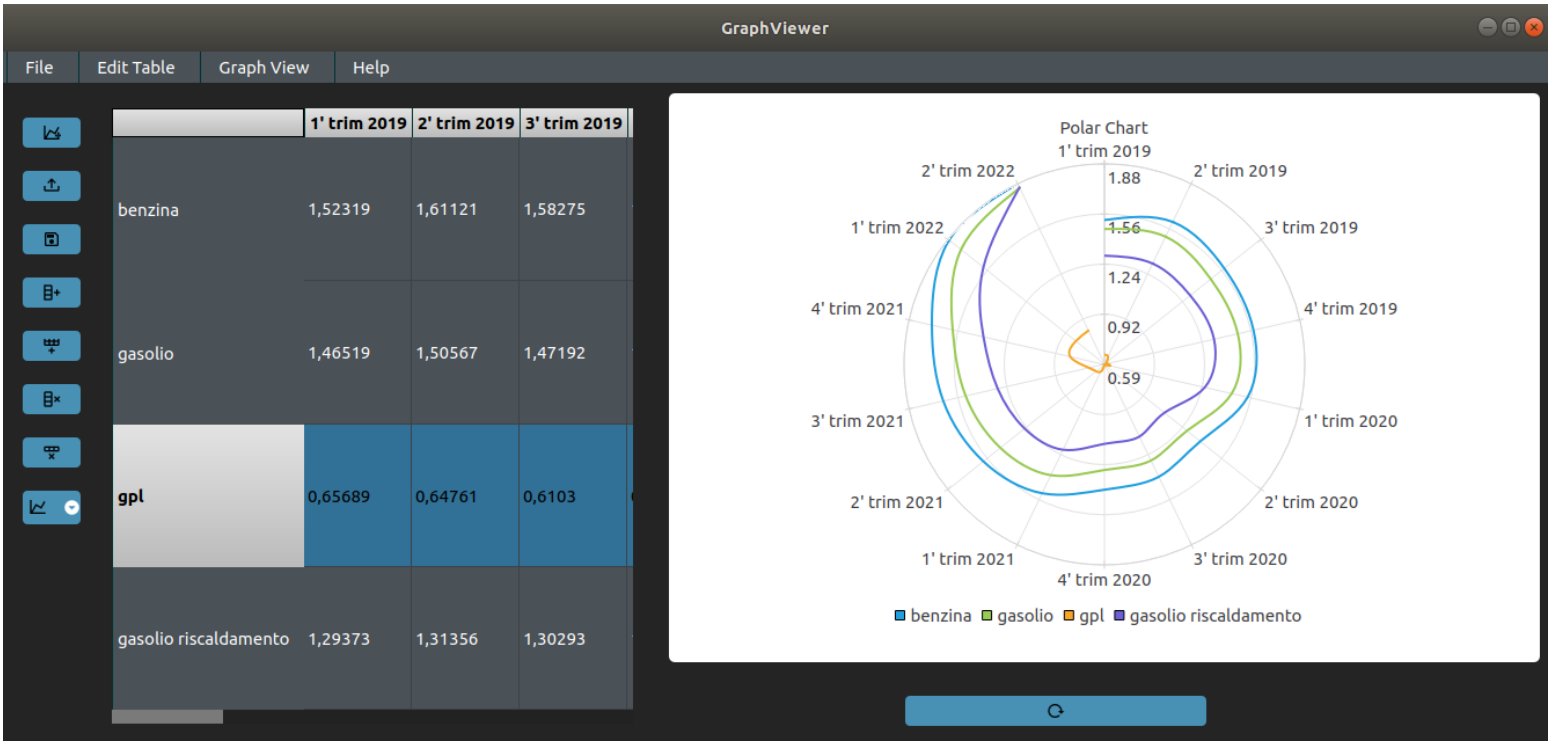
GraphViewer è un software che permette di **creare**, **modificare**, **salvare** e **visualizzare** fino a sei tipi di **grafici** diversi, tra cui :

- Line Chart;
- Bar Chart;
- Pie Chart;
- Area Chart;
- Scatter Chart;
- Polar Chart;

I dati rappresentati nei grafici vengono gestiti tramite l'uso di una **tabella**. GraphViewer offre varie funzionalità, tra cui :

- **Creazione** di un nuovo progetto con una tabella di dimensione personalizzata;
- **Inserimento** e **rimozione** di righe e/o colonne;
- Inserimento e **modifica** dei dati nella tabella;
- Modifica delle intestazioni della tabella;
- **Visualizzazione** di uno dei cinque grafici con input della tabella creata;
- **Importazione** ed **esportazione** dei dati in formato JSON;

1.b - Screenshot rappresentativo del software



2 - Descrizione delle gerarchie dei tipi

2.a - Il Modello

La classe DataModel estende la classe QAbstractTableModel di Qt.
DataModel rappresenta la struttura dati del progetto e ne consente la manipolazione tramite i metodi pubblici implementati.
I dati vengono memorizzati in un vettore di float a due dimensioni denominato “tableData” e due vettori di QVariant denominati rispettivamente “rowHeader” e “ColumnHeader”.

All’interno della classe DataModel inoltre vengono definite le funzioni readJson e writeJson per l’esportazione e importazione della struttura dati.

2.b - BaseChart

La classe astratta BaseChart estende la classe QChart di QtCharts, essa viene ereditata nell’implementazione delle classi concrete :

- LineChart : definisce il grafico a linee tramite la classe QLineSeries, si utilizza QCategoryAxis per i label delle colonne sull’asse X e QValueAxis per i valori dell’asse Y;

- BarChart : definisce il grafico a barre tramite la classe QBarSeries e la classe QBarSet, si utilizza QBarCategoryAxis per i label delle colonne sull’asse X e QValueAxis per i valori dell’asse Y;

- PieChart : definisce il grafico a torta tramite la classe QPieSeries e la classe QPieSlice;

- AreaChart : definisce il grafico ad area tramite la classe QAreaSeries e la classe QLineSeries, si utilizza QCategoryAxis per i label delle colonne sull’asse X e QValueAxis per i valori dell’asse Y;

- ScatterChart : definisce il grafico a dispersione tramita la classe QScatterSeries, si utilizza QCategoryAxis per i label delle colonne sull’asse X e QValueAxis per i valori dell’asse Y;

- PolarChart : definisce il grafico polare tramite la classe QSplineSeries, si utilizza QCategoryAxis per i label delle colonne sull’asse X e QValueAxis per i valori dell’asse Y;

2.c - View

La classe GraphViewer estende la classe QWidget e rappresenta l'interfaccia del progetto. GraphViewer contiene i componenti grafici per la creazione e visualizzazione della finestra principale del software.
All'interno sono stati dichiarati vari campi dati tra cui i più importati :

- Controller il cui compito è di collegare i vari pulsanti con gli adiacenti metodi;
- QGridLayout il cui compito è di gestire il layout principale e contenere successivi widget e layout come :
    - un layout di vari QMenu contenente il menu principale del progetto, posizionati nella prima riga del QGridLayout;
    - un layout di vari QPushButton e QComboBox posizionato nella seconda riga e prima colonna del QGridLayout;
    - QTableView il quale permette la visualizzazione del DataModel in una tabella posizionato nella seconda riga e seconda colonna del QGridLayout;
    - QChartView il quale permette la visualizzazione del QChart in una tabella posizionato nella seconda riga e seconda colonna del QGridLayout;

2.d - Controller

La classe Controller estende QObject e ha il compito di gestire i collegamenti dei vari pulsanti della classe GraphViewer.
Si occupa inoltre di controllare i dati in input tramite interfaccia grafica.

2.e - Doc e About us

Le classi Doc e About us estendono QWidget con il compito gestire la visualizzazione corretta della documentazione e le informazioni di contatto.

3 - Descrizione dell'uso delle chiamate polimorfe

3.a - BaseChart

La classe astratta BaseChart contiene al suo interno un campo dati DataModel protected che punta al modello che si deve rappresentare tramite grafico. Inotre contiene metodi virtual che vengono implementate nelle classi ..Chart elencate sopra nel paragrafo 2.b.
Tra cui i seguenti :

- virtual void dataToSeries(QChart* chart)=0;
    - Ottiene la tabella del campo DataModel con le rispettive intestazioni e costruisce le serie e le assi per il grafico dato da implementare.
- virtual QChart* toChart()=0;
    - Crea il grafico QChart, invoca dataToSeries(Qchart* chart) per ottenere ed inserire il contenuto per poi ritornare un QChart.

- virtual ~BaseChart();
    - Distruttore virtual implementato da ogni classe che implementa la classe BaseChart.

4 - Descrizione dei formati di input e output

4.a - JSON (JavaScript Object Notation)

Il progetto permette di salvare su un file .json la tabella utilizzata per visualizzare i grafici, inoltre permette di importare un file .json con all'interno la tabella da caricare.
La struttura JSON è la seguente :

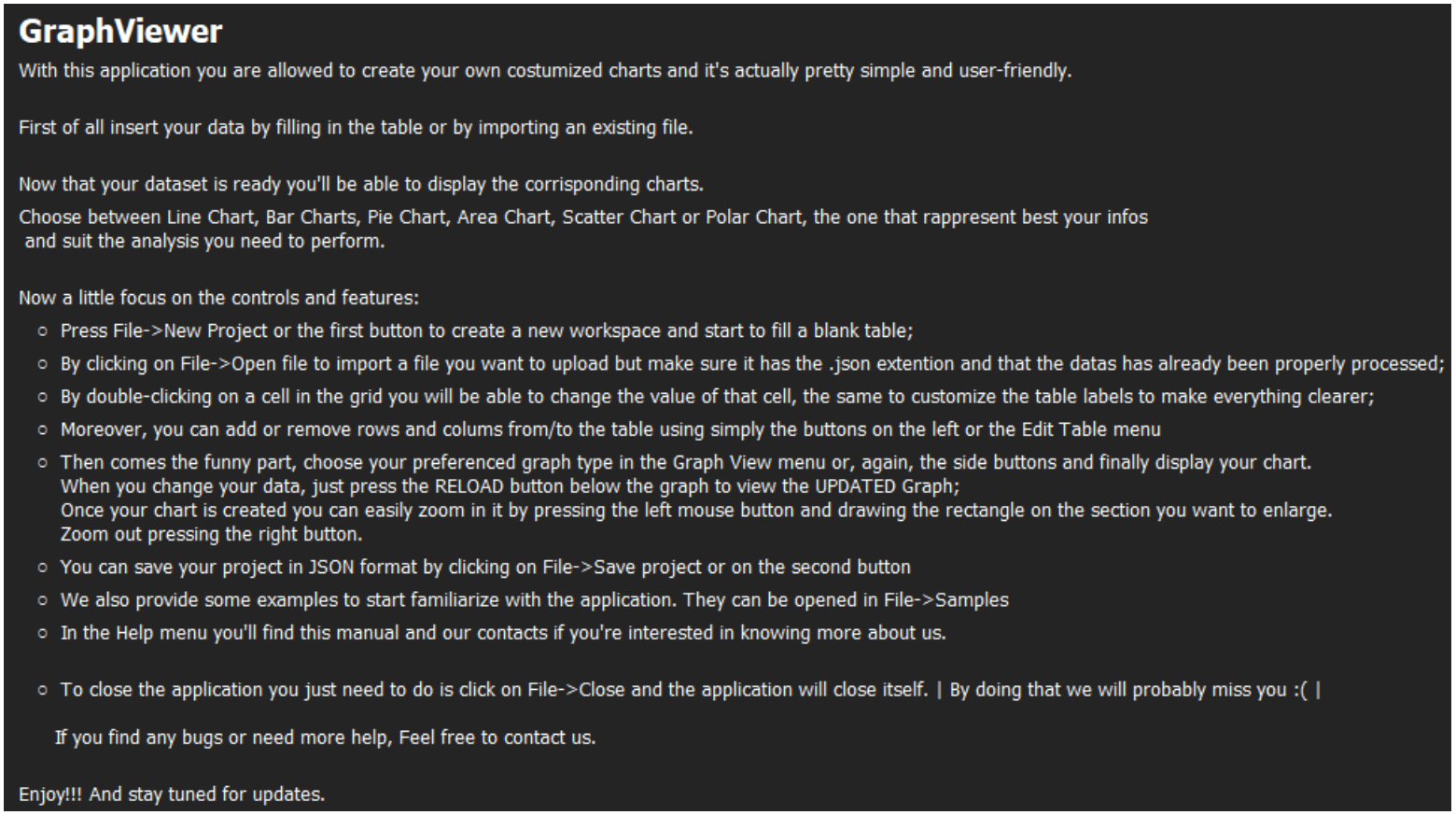
```
{
   "columnHeader": [
      "Intestazione prima colonna",
      "...",
      "Intestazione ultima colonna"
   ],
   "rowHeader": [
      "Intestazione prima riga",
      "...",
      "Intestazione ultima riga"
   ],
   "tableData" : [
      [
         Primo valore float della prima colonna,
         …,
         Ultimo valore float della prima colonna
      ],
       [
         Primo valore float della n colonna,
         …,
         Ultimo valore float della n colonna
      ],
       [
         Primo valore float dell'ultima colonna,
         …,
         Ultimo valore float dell'ultima colonna
      ]
   ]
}
```

Nota bene: columnHeader e rowHeader accettano sia valori numeri che stringhe mentre i valori della tableData devono essere di tipo float (esempio 2.0002).
Per un import di JSON corretto columnHeader e rowHeader devono essere della stessa dimensione rispettivamente per le colonne e le righe della tableData.

5 - Manuale utente della GUI

5.a - Descrizione

Il software è stato progettato per essere il più intuitivo possibile grazie all'uso di immagini per descrivere le funzioni e il design dell'interfaccia molto semplificato.
All'interno del menù è presente la sezione Help->Documentation che apre la seguente finestra :



6 - Istruzioni di compilazione ed esecuzione

6.a - Prerequisiti

Per compilare il file occorre aver installato :

- una versione di Qt (5.9.5) con libreria Qt Charts;
- compilatore GNU g++ (7.3)

6.b - Istruzioni

Su una shell localizzata nella cartella del progetto è possibile compilare ed eseguire il progetto tramite i seguenti comandi:

- **qmake ./GraphViewer.pro**
- **make**
- **./GraphViewer**

7 - Descrizione delle ore richieste per task

Task	Ore
Analisi del problema	2
Progettazione Modello e GUI	8
Codifica Modello	8
Implementazione View e Controller	10
Codifica BaseChart, BarChart	3
Implementazione Read e Write to JSON	4
Debug e HotFix	8
Testing	4
Stesura della documentazione	1
Relazione	3
Totale	51

La realizzazione del progetto ha impiegato 51 ore, poco più delle ore previste per via di vari testing e debugging mirati al controllo della stabilità del progetto così da aumentarne l'efficienza e lo spessore.

8 - Suddivisione del lavoro progettuale

Matricola	Task
Massimo Chioru 2013449	Progettazione Modello e GUI
	Codifica Modello
	Implementazione View e Controller
	Codifica BaseChart, BarChart
	Implementazione Read e Write to JSON
	Debug e HotFix
	Stesura della documentazione
Elena Fabris 2008072	Progettazione Modello e GUI
	Codifica View e Controller
	Codifica PieChart, LineChart, AreaChart, ScatterChart e PolarChart
	Implementazione Stile e Grafica
	Scrittura Sample JSON
	Stesura della documentazione