PROGRAMMAZIONE AD OGGETTI
A.A. 2022/2023

qaria

Relazione al progetto di Programmazione a Oggetti

Giovanni Filippini - 2052784

1 Introduzione

Nel contesto degli ultimi anni, i fenomeni meteorologici avversi sono diventati sempre più frequenti e catastrofici, tanto da causare danni significativi al loro manifestarsi. È pertanto cruciale sviluppare strumenti in grado di affrontare efficacemente tali eventi. In questo contesto, l'applicativo *qaria* è stato sviluppato per tenere traccia, catalogare e consultare gli eventi meteorologici avversi, focalizzandosi sulle tre tipologie principali, quali pioggia, grandine e neve, e sulle loro caratteristiche specifiche.

L'obiettivo principale di *qaria* è offrire una soluzione di facile comprensione e utilizzo, che consenta agli utenti di comprendere chiaramente gli eventi meteorologici avversi presenti nella zona. L'applicativo offre un'interfaccia intuitiva e la possibilità di filtrare facilmente i fenomeni atmosferici in base alla tipologia o alla pericolosità, sfruttando alla base il polimorfismo.

Nello specifico, si è cercato di utilizzare il polimorfismo in modo naturale, realizzando con questo l'implementazione degli eventi atmosferici. Grazie a questo, *qaria* è in grado di gestire in modo dinamico e flessibile i diversi tipi di eventi meteorologici, adattando il codice alle specifiche caratteristiche dei fenomeni climatici come pioggia, grandine e neve, visualizzando le caratteristiche e l'immagine delle precipitazioni.

Un aspetto fondamentale di *qaria* è la possibilità di analizzare i *trend* meteorologici avversi nel corso del tempo. La catalogazione e la visualizzazione chiara degli eventi consentono di individuare facilmente le tendenze passate e di fare previsioni specifiche, grazie alle opzioni di filtraggio presenti.

L'applicativo può quindi adattarsi, grazie alla sua facilità d'uso e la sua capacità di gestire dati specifici per tipo di eventi, offrendo un'esperienza utente completa per affrontare concretamente i fenomeni meteorologici.

2 Descrizione del modello

2.1 Gerarchie ed ereditarietà

Il modello parte dalla classe *AtmosphericEvent* che rappresenta tutte le informazioni comuni agli eventi. Le tre sottoclassi *Snow*, *Rain* ed *HailStorm* rappresentano gli eventi atmosferici veri e propri, mediante una relazione **is-a**, ed applicano i parametri relativi all'evento atmosferico.

Sono presenti due tipologie di modello: **DataModel** e **AtmosphericEventModel**. Il primo viene utilizzato per la gestione dei dati all'interno dei file CSV, il secondo per filtrare gli eventi e il loro tipo. Inoltre, sia le classi di gestione dei modelli e dati che le viste estendono classi preesistenti al framework Qt per garantire coerenza ai metodi presenti. Sono inoltre presenti delle classi delegate che applicano il noto *design pattern Visitor* nell'utilizzo della visualizzazione dei dati attinenti allo specifico evento atmosferico, rendendo possibile la visualizzazione specifica in base al contesto richiesto.

Le classi proteggono le proprie informazioni dall'accesso esterno, consentendo l'accesso protetto delle variabili che permettono la gestione a livello grafico degli eventi atmosferici da visualizzare in base al tipo specifico come descritto.

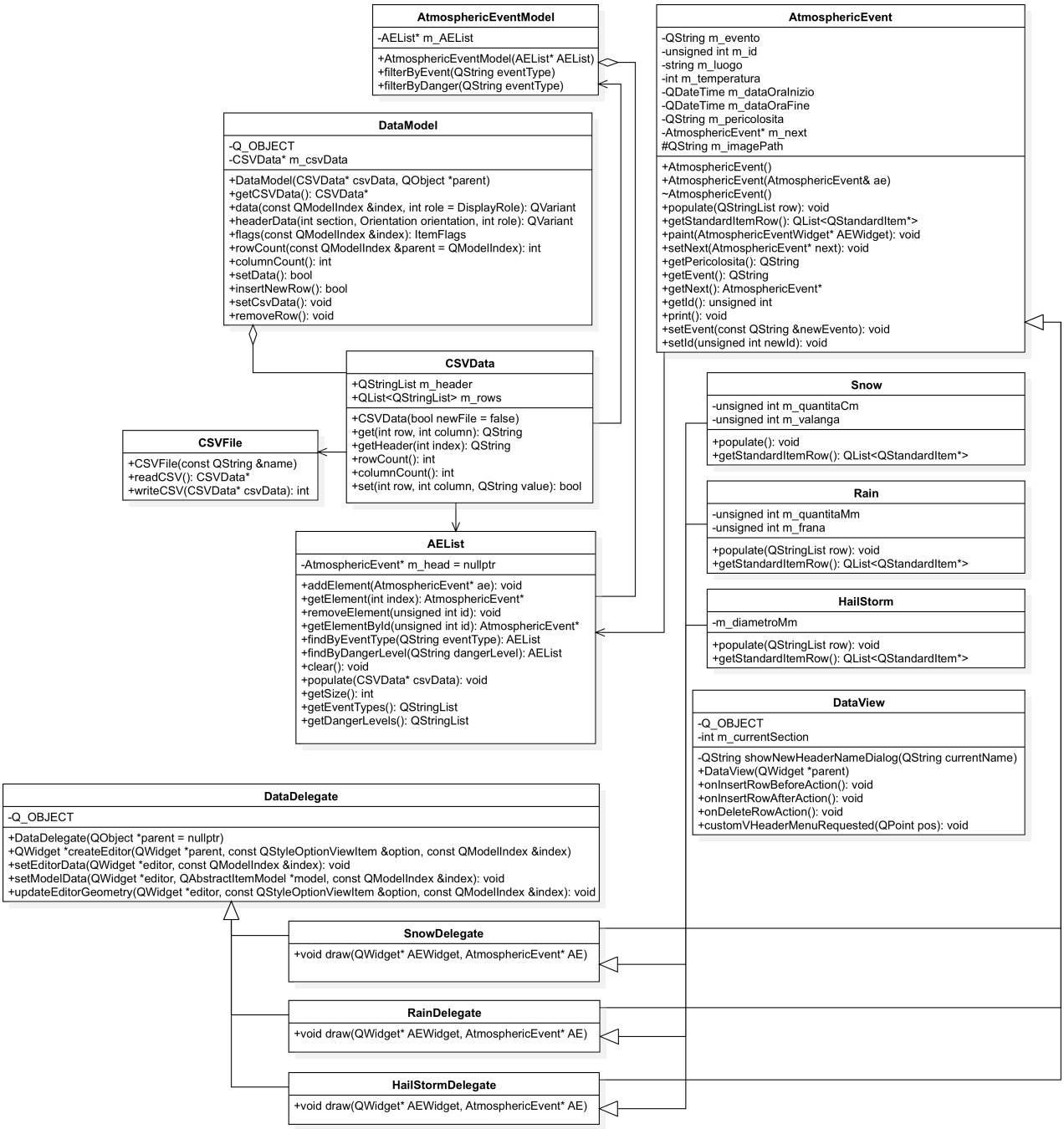

Figura 1: Diagramma completo delle classi progettuali

Anziché avere quattro file separati per rappresentare la classe padre e le relative sottoclassi, esse sono dichiarate dentro il file *atmophericevent.cpp*, senza comprometterne il funzionamento o l'ereditarietà delle stesse.

Tuttavia, ciascuna classe ha un suo comportamento indipendente da un punto di vista logico dalle altre, al fine di separare logicamente le responsabilità di ciascuna e gestire gli eventi in maniera virtuale ed estendibile.

3 Polimorfismo

Nella realizzazione del progetto *qaria* , l'utilizzo non banale del polimorfismo si evidenzia principalmente tra la classe padre *AtmosphericEvent* e le sue classi figlie. In particolare, è stato impiegato per la creazione del *widget* che visualizza le informazioni dell'evento, che viene aperto graficamente come menù a tendina e come tabella, consentendo in entrambi i casi la precisa modifica del tipo presente.

L'utilizzo del polimorfismo avviene esclusivamente durante la fase di visualizzazione. Quando si espande la tendina in base all'evento selezionato, vengono mostrati i dettagli appropriati per lo specifico; lo stesso vale per il *widget* che visualizza le informazioni corrette per quello selezionato.

Inoltre, anche la gestione del file CSV prevede l'utilizzo di metodi estensibili per la gestione delle sue componenti di dati e per i dati mostrati successivamente alle viste in base ai cambiamenti fatti sul modello.

È stata prevista la creazione di una gerarchia di classi delegate aventi come classe padre *DataDelegate* al fine di permettere la comunicazione e il collegamento tra i dati aggiornati nel modello e quanto viene mostrato all'interno del *widget*. Per quest'ultima, possiamo specificare una serie di classi figlie delegate, facenti riferimento a ciascun tipo di evento atmosferico (nello specifico, *SnowDelegate*, *RainDelegate* ed *HailStormDelegate*) con lo scopo di gestire, tramite apposito metodo virtuale di disegno *draw*, l'aggiornamento dei dati provenienti dal modello nella vista specifica all'evento richiesto.

3.1 Polimorfismo non banale

Per rendere l'utilizzo del polimorfismo non banale, la classe *DataDelegate* implementa un metodo chiamato *draw*, il quale disegna un *widget* personalizzato quando viene selezionato un evento dalla lista. Questo metodo consente di mostrare non solo i valori comuni a tutte le classi di eventi, ma anche quelli specifici a ciascuna classe. Inoltre, per evidenziare l'utilizzo del polimorfismo, viene visualizzata un'icona che rappresenta l'evento atmosferico utilizzando l'attributo *m_imagePath* presente in *AtmosphericEvent*.

Questo approccio sfrutta il polimorfismo in modo significativo, consentendo a ciascuna sottoclasse di *AtmosphericEvent* di definire il proprio delegate per visualizzare le informazioni e l'icona specifiche dell'evento. Ciò offre una maggiore flessibilità e estensibilità al sistema, in quanto nuove classi di eventi possono essere facilmente aggiunte e gestite in modo dinamico dal *widget* di visualizzazione.

4 Persistenza dei dati

Per il salvataggio delle informazioni viene utilizzato il formato CSV, in modo tale da poter permettere la condivisione degli eventi con altri utenti; infatti è possibile importare o creare dei nuovi file da zero, in modo da catalogare nel miglior modo possibile i dati e quindi gestire le loro informazioni.

Un esempio della struttura dei dati è resa disponibile mediante il file *dati.csv* allegato nella consegna, che può essere caricato nel programma facilmente selezionando l'opzione apposita.

4.1 Creazione, modifica e cancellazione degli oggetti

Per creare un nuovo oggetto, è sufficiente fare clic destro su una riga e selezionare dove inserirla, in particolare sopra o sotto quella selezionata.
La cancellazione è possibile usando lo stesso ragionamento. Se seleziono la riga, posso decidere se eliminarla

selezionando dal menù a comparsa.
Per la modifica delle informazioni, è sufficiente cliccare due volte sulle celle. In questo caso, occorre sottolineare la presenza un bug di Qt su macOS poiché, quando i due *QComboBox* utilizzati per selezionare l'evento e la pericolosità vengono invocati, rimane visibile sotto la precedente scelta. Questo comportamento non si presenta su altre piattaforme.

4.2 Contenitore per oggetti della gerarchia

Si è deciso di sviluppare una struttura *linked list* da utilizzare come contenitore della gerarchia, data la sua estendibilità per nuove operazioni di calcolo. In questo caso, prima di essere salvati nel file CSV, i vari eventi atmosferici sono salvati in locale sulla *linked list* e poi, una volta che l'utente decide di salvare i dati, vengono scritti sul file CSV. Inoltre, nel contenitore sono stati implementati dei metodi personalizzati per la ricerca, cancellazione, modifica ed aggiunta degli oggetti.

5 Funzionalità e GUI

5.1 Funzionalità

Le funzionalità implementate sono, per semplicità, suddivise in due macro-categorie: strutturali, ovvero quelle che compongono il cuore del programma, ed estetiche, quelle che invece ne permettono un facile utilizzo.

5.1.1 Funzionalità strutturali

- Creazione, apertura, salvataggio e sovrascrittura di un file CSV.
- Ricerca per tipologia e pericolosità dell'evento.
- Aggiunta degli eventi atmosferici.
- Eliminazione dei singoli eventi.
- Filtri per i risultati

5.1.2 Funzionalità estetiche

- Barra del menù.
- Controllo della presenza di modifiche non salvate.
- Utilizzo di icone nei pulsanti.
- Dimensionamento minimo della finestra.
- Vista per dettaglio degli eventi.
- Sono inoltre implementate le seguenti scorciatoie da tastiera (mostrate anche nelle voci del menù):

Azione	**Scorciatoia (macOS)**	**Scorciatoia (Windows)**
Nuovo documento	Cmd + Shift + N	Ctrl + Shift + N
Salva	Cmd + Shift + S	Ctrl + Shift + S
Salva come...	Cmd + Shift + S	Ctrl + Shift + S
Cambia vista	Cmd + Shift + V	Ctrl + Shift + V
Apri...	Cmd + Shift + O	Ctrl + Shift + O

Le combinazioni funzionano correttamente su macOS, mentre su Windows e Ubuntu hanno un funzionamento parziale.

5.2 Guida GUI

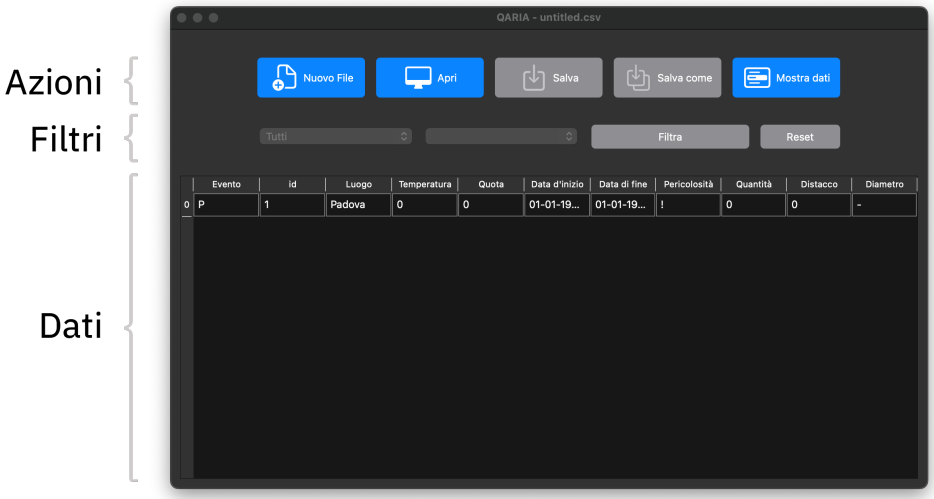

Figura 2: Come si presenta l'applicazione appena aperta

L'applicazione è costituita da tre parti essenziali. La prima contiene le *Azioni*, che sono i comandi principali come creazione, apertura, salvataggio, etc. All'inizio i pulsanti per effettuare il salvataggio sono disattivati, poiché si attivano solo quando vengono effettuate delle modifiche al file. Il pulsante *Mostra dati* permette di passare dalla modifica dei dati ad una vista dettagliata degli eventi.

La seconda parte è dedicata ai *Filtri*, attivabili solamente quando si stanno vedendo in dettaglio i dati. Si può filtrare per tipologia di evento (*Pioggia*, *Neve* e *Grandine*) e per pericolosità. È inoltre presente un pulsante per il reset dei filtri.

La parte inferiore dell'applicazione è occupata dalla tabella in cui è possibile modificare, aggiungere o eliminare dati. Essa viene nascosta quando viene premuto il pulsante *Mostra dati*, per lasciare spazio alla vista dettagliata degli eventi.

Appena l'applicazione viene aperta, vi sono già dei dati. É possibile iniziare a modificarli o aggiungerne di nuovi fin da subito.

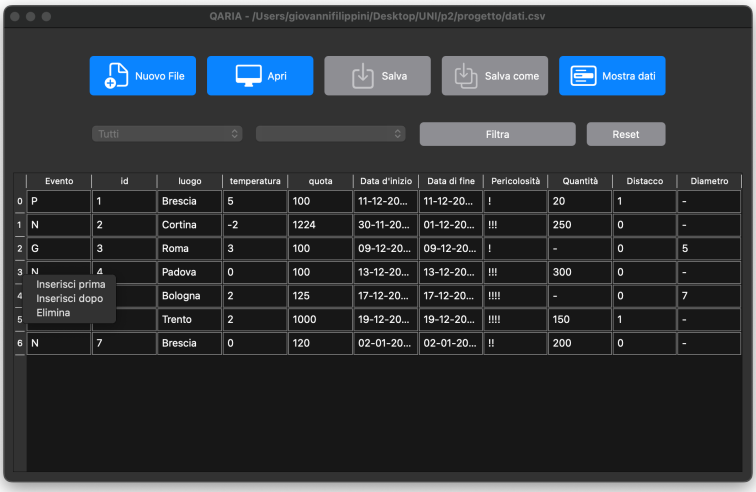

Figura 3: Come si presenta la tabella quando si vanno a modificare i dati e ad aggiungere una nuova riga

Si può scegliere di creare un nuovo file o iniziare con uno già presente sul proprio dispositivo. Quando viene modificato un file, si ha l'opzione di salvarlo o crearne uno nuovo. Se si decide di crearne uno nuovo, verrà chiesta la posizione in cui si desidera salvarlo. Per vedere se un file è modificato viene visualizzato il simbolo * accanto al percorso del file.

Inoltre, si può aggiungere o eliminare un dato facendo clic destro sul numero della riga. Questo farà apparire un menù dove si può scegliere se aggiungere una nuova riga prima o dopo quella selezionata, oppure eliminarla.

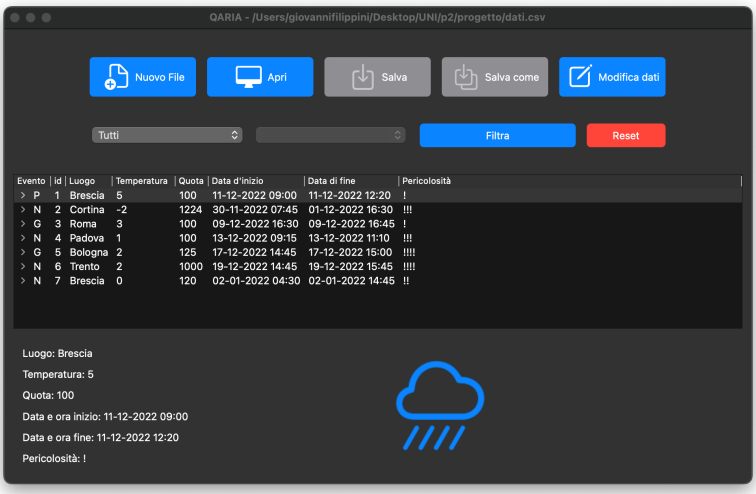

Figura 4: Come appare la vista dettagliata degli eventi

Per passare alla visione dei dati, basterà premere su *Mostra dati*. Una volta in questa sezione, si può premere sulla freccia sulla destra per vedere i dati associati ad un evento specifico, mentre premendo sull'evento in sé comparirà nella parte sottostante un piccolo *widget* che mostrerà l'evento in dettaglio, con un icona per indicare di che tipo di evento si tratta.

6 Considerazioni finali

6.1 Rendicontazione ore

Attività	**Ore effettive**
Studio e progettazione	5
Sviluppo del codice del modello	20
Studio del framework Qt	15
Sviluppo del codice della GUI	6
Test e debug	9
Stesura della relazione	4
Modifiche apportate dalla precedente consegna	7
Totale	**66**

Il monte ore è stato superato in quanto lo studio approfondito del framework Qt, necessario per lo sviluppo dell'applicativo, e l'attività di debug hanno richiesto un tempo maggiore rispetto a quanto inizialmente previsto. Rispetto alla consegna precedente, è stata aggiunta la percentuale di ore di lavoro impiegate per le modifiche previste da precedente correzione.

6.2 Ambiente di Sviluppo

Lo sviluppo principale è stato effettuato sulla piattaforma:

MacBook Pro 2017 (MacBook Pro 14,3)

macOS Ventura 13.3.1 (a)

clang 14.0.3

Qt 6.4.3

Il programma è stato inoltre testato sulla macchina virtuale resa disponibile e su Windows (Windows 10 Pro, versione 22H2).

6.3 Modifiche apportate dalla precedente consegna

Sono state effettuate le modifiche necessarie in modo da disaccoppiare il modello dalla sua rappresentazione grafica, garantendo così un codice più flessibile e maggiormente mantenibile. Nel dettaglio, è stata spostata dalla classe *AtmosphericEvent* a *DataDelegate* la funzione *paint*, rinominata in *draw*, e i parametri *QLabel* e *QGridLayout*, come consigliato nel feedback.

Le due funzioni accessorie *populate* e *getStandardItemRow* non sono state spostate in quanto, dopo un confronto, è stato stabilito che non sono legate all'interfaccia grafica. Per questo motivo, è stata introdotta una maggiore separazione logico-grafica attraverso l'utilizzo della specificata gerarchia di classi delegate.

Nella GUI sono stati ridisegnati i pulsanti, aggiungendo un colore per rappresentare al meglio gli stati e le azioni collegate (come ad esempio il rosso per le azioni distruttive). Per le *QComboBox* è stato deciso di lasciarle con lo stile del sistema operativo per evitare glitch grafici e problemi di leggibilità dei contenuti. Anche alle icone che rappresentano gli eventi atmosferici è stata data una tonalità di blu leggermente più brillante, per risultare maggiormente visibili, oltre ad essere stato ridotto il peso dei vettoriali di circa l'80%.
Inoltre, è stata migliorata la gestione del ridimensionamento della finestra, partendo dalle azioni che sono centrate e non vanno ad occupare tutta la larghezza della finestra e anche la tabella si adatta dinamicamente allo spazio disponibile e non è più in parte nascosta quando si ha una finestra di piccole dimensioni.

La relazione è stata riscritta rendendola più accessibile utilizzando il comando *section*, vi è una nuova e migliore rappresentazione delle classi nello schema UML e il rifacimento della guida per la GUI con l'inclusione di immagini. È stato infine aggiunto un maggiore spazio nell'interlinea per migliorare la leggibilità.