Progetto Grammer - Monochat

Stafa Diego, Matricola 1226285

8 Settembre 2021

1 - Descrizione del progetto

Grammer - Monochat è un progetto atto ad emulare una generica applicazione di messaggistica, nella quale ogni utente singolarmente può effettuare il login o registrarsi per creare chatroom ed inviare messaggi ad altri utenti. I messaggi possono essere di vario tipo e in base a ciò diverse operazioni possono essere effettuate su di essi, olre a permettere l'ordinamento dei messaggi, le room implementano un sistema di filtraggio dei messaggi basato su diversi parametri, quali data, tipo e testo contenuto. Grammer consente anche di rispondere ad altri messaggi, eliminare i propri oppure inoltrarli in altre room e nel caso di invio di un'immagine viene viene fornita la possibilità di visualizzarne una preview.

2 - Descrizione delle gerarchie di tipi

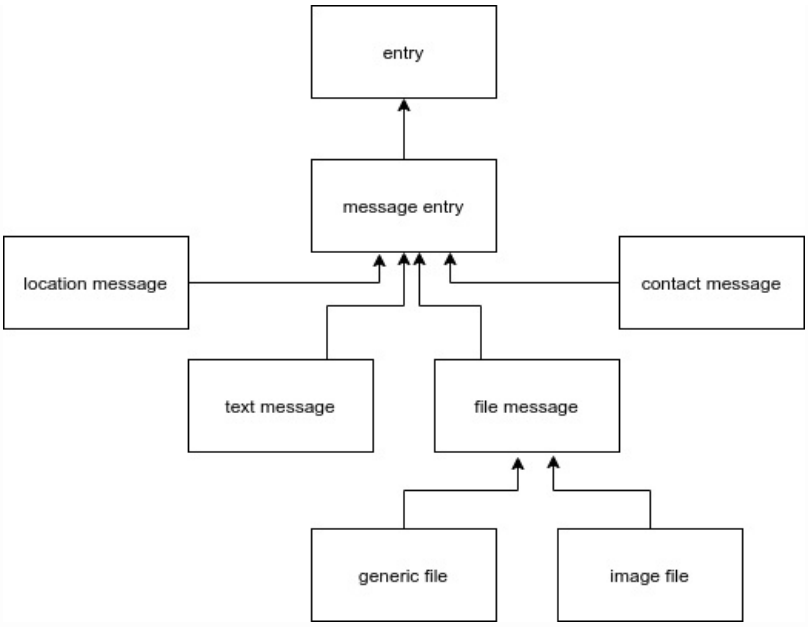
Il modello presenta 2 gerarchie principali:

- la gerarchia delle entry
- la gerarchia dei db

entrambe le gerarchie dipendono solamente dalla STL.

2.1 - Gerarchia delle entry

Questa gerarchia costituisce gli oggetti base su cui una room opera



entry

è il tipo astratto che una room gestisce, è definito da uno unix timestamp di invio ed un identificativo numerico, va a definire un'interfaccia su: * operatori relazionali, basata sugli ID e la data di invio * rappresentazione testuale (to_string) * analisi del tipo

message entry

rappresenta un altro tipo astratto, è un'entry che definisce messaggio, quindi inviabile da un utente, matiene uno stato del messaggio che indica se questo è stato modificato od inoltrato, contiene inoltre un riferimento ad una entry che rappresenta l'oggetto/evento a cui si vuole rispondere

text message

rappresenta un semplice messaggio testuale, può essere costruito con un'altra entry grazie all'interfaccia to_string

location message

rappresenta una posizione geografica

contact message

rappresenta un contatto

file message

rappresenta un file generico, definito da un percorso file e da un peso in byte, implmenta un suo ordinamento

image file

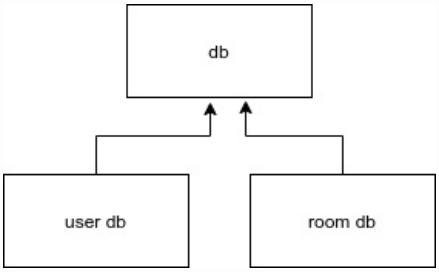
rappresenta un immagine, definita quindi dalle dimensioni

generic file

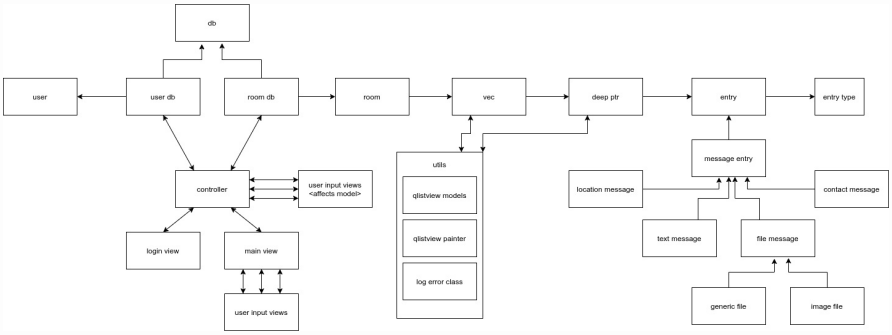
un file generico, concretizzazione di file message

2.2 - Gerarchia di db

Questa piccola gerarchia serve a mantenere un database di utenti e room, la base definisce un interfaccia comune alle sottoclassi per le varie operazioni



2.3 Struttura complessiva



3 - Polimorfismo

il polimorfismo e l'uso di chiamate polimorfe viene sfruttato in diversi casi:

- **to_string**: è il metodo virtuale che consente ad ogni entry di essere rappresentata come stringa, utilizzato dal painter per la visualizzazione dei messaggi a schermo.
- **get_type**: ritorna il tipo della sottoclasse concreta, utilizzato dal painter per disegnare in modo diverso a seconda dellìoggetto su cui lavora, ad esempio per disegnare le icone.
- **is_message**: determina se l'entry è un messaggio o no, usato per disegnare i messaggi, per determinare se un'entry è inoltrabile/modificabile
- **copy_forward**: ritorna un messaggio che rappresenta la copia inoltrata di un messaggio.
- **operatori relazionali**: consentono l'ordinamento di diversi tipi di entry.
- **clone**: viene utilizzato ogni qual volta il deep_ptr necessita di fare un'assegnazione profonda, ad esempio quando il *vec* (container personale)

deve essere ridimensionato e deve fare una copia degli elementi.
- **distruttore**: viene usato dal container personale quando va ad deallocare gli oggetti dopo un ridimensionamento, viene ovviamente anche usato per tutte le distruzioni in generale di puntatori polimorfi

4 - Manuale utente

login window:

- per registrarsi è sufficente inserire uno username univoco, i campi dati devono essere lunghi almeno 4 caratteri
- per effettuare il login bisongna ovviamente inserire la coppia username/password corretta, nel progetto sono state inseriti utenti random, è possibile fare il login con la coppia **diego/diego**

main window:

- in assenza di rooms l'applicazione non fa molto, viene quindi chiesto di crearne e selezionarne una prima di qualsiasi operazionedd attraverso il pulsante **new room**
- dopo aver selezionato una room si entra in modalità **selected room** ed è possibile iniziare ad inviare messaggi ed eseguire operazioni sulla room
  - filtraggio per tipo/testo/data
  - ordinamento globale della room
- dopo aver selezionato un messaggio si entrà in modalità di **selected entry** ed è possibile rispondere alle entry ed eseguire operazioni su di esse
  - image file: mostrerà un pulsante per la preview dell'immagine
  - non è possibile eliminare messaggi altrui
- è possibile fare il *logout* attraverso il pulsante **profile**, fare il login con un altro utente mostrerà la lista dei messaggi aggiornati, fare *quit* terminerà l'applicazione ed eliminerà ogni messaggio

5 - Compilazione ed esecuzione

nel caso il file *grammer.pro* sia presente, il progetto può essere compilato ed eseguito con i seguenti comandi:

```
qmake
make
./grammer
```

6 - Tempo richiesto

il tempo richiesto per le varie fasi è stato così suddiviso:

- analisi preliminare: **2h**
- progettazione del modello: **9h**

- progettazione della gui: **3h**
- apprendimento QT: **20h**
- codifica del modello: **5h**
- codifica della gui: **15h**
- debugging e testing: **6h**

totale: 50h + 10h

l'overtime è stato causato dal painting e formattazione dei messaggi a schermo, in quanto qualcosa di mai affrontato prima d'ora.

7 - Ambiente di sviluppo

- Arch Linux / Linux 5.10.61-1-lts
- Qt version 5.15.2
- g++ (GCC) 11.1.0
- valgrind-3.17.0

Note di progetto

la classe entry ed message_entry sembrerebbero essere ridondanti tra loro, ma ho preferito separarle in quanto a mio parere è plausibile un'estensione della gerarchia che ammetta un tipo diverso dai messaggi

il painting dei messaggi a schermo è stato più arduo del previsto ed è presente un glitch grafico quando si inviano messaggi troppo lunghi, ciò è dato dall'altezza fissata delle righe della QListView che non è stata resa variabile a causa della mia ristretta conoscenza della libreria, nonostante ciò questo glitch non determina un malfunzionamento logico del programma