oggetto	Relazione progetto Programmazione a Oggetti
autore	Zaupa Riccardo, mat. 2034303
titolo	CineMatrix

Introduzione

CineMatrix è un sistema di gestione delle prenotazioni di biglietti per spettacoli cinematografici. La piattaforma permette agli utenti di creare facilmente prenotazioni, tra una vasta selezione di film, date e orari disponibili. Inoltre, offre la comodità di poter cancellare una prenotazione nel caso si cambi idea.

Una delle caratteristiche distintive di *CineMatrix* è la sua rappresentazione grafica delle sale cinematografiche, che mostra in modo intuitivo i posti disponibili e i relativi prezzi. Ogni tipo di posto viene visualizzato in modo differente. Una volta effettuata una prenotazione, il sistema garantisce che lo stesso posto non possa essere scelto per lo stesso spettacolo, assicurando così una distribuzione equa dei posti tra gli spettatori.

Inoltre, *CineMatrix* fornisce informazioni dettagliate su ogni prenotazione, consentendo agli utenti di visualizzare i posti selezionati nelle prenotazioni precedenti. Questa funzionalità offre un utile riferimento per gli utenti e facilita la pianificazione delle future prenotazioni.

Un aspetto importante del sistema è la varietà di tipologie di posti disponibili. Ci sono i posti poltrona comfort, che offrono un prezzo nella media per un'esperienza di visione piacevole. I posti a prezzo scontato consentono ai visitatori di partecipare all'esperienza cinematografica a un costo ridotto, mentre i posti VIP offrono caratteristiche aggiuntive per un'esperienza di lusso.

Il principale punto di forza dell'applicazione è l'interfaccia intuitiva e la sua architettura scalabile.

Ho scelto di lavorare a questo progetto perché la prenotazione dei posti in sale cinematografiche è un'esperienza comune per me, e volevo sfruttare la mia conoscenza e interesse per il cinema nell'implementazione di un sistema simile a quelli esistenti come i siti di Uci Cinemas, The Space, ecc.…. Ho poi pensato che i diversi tipi di posti fossero un buon pretesto per utilizzare il polimorfismo in maniera non banale, come richiesto nelle specifiche.

Descrizione del modello

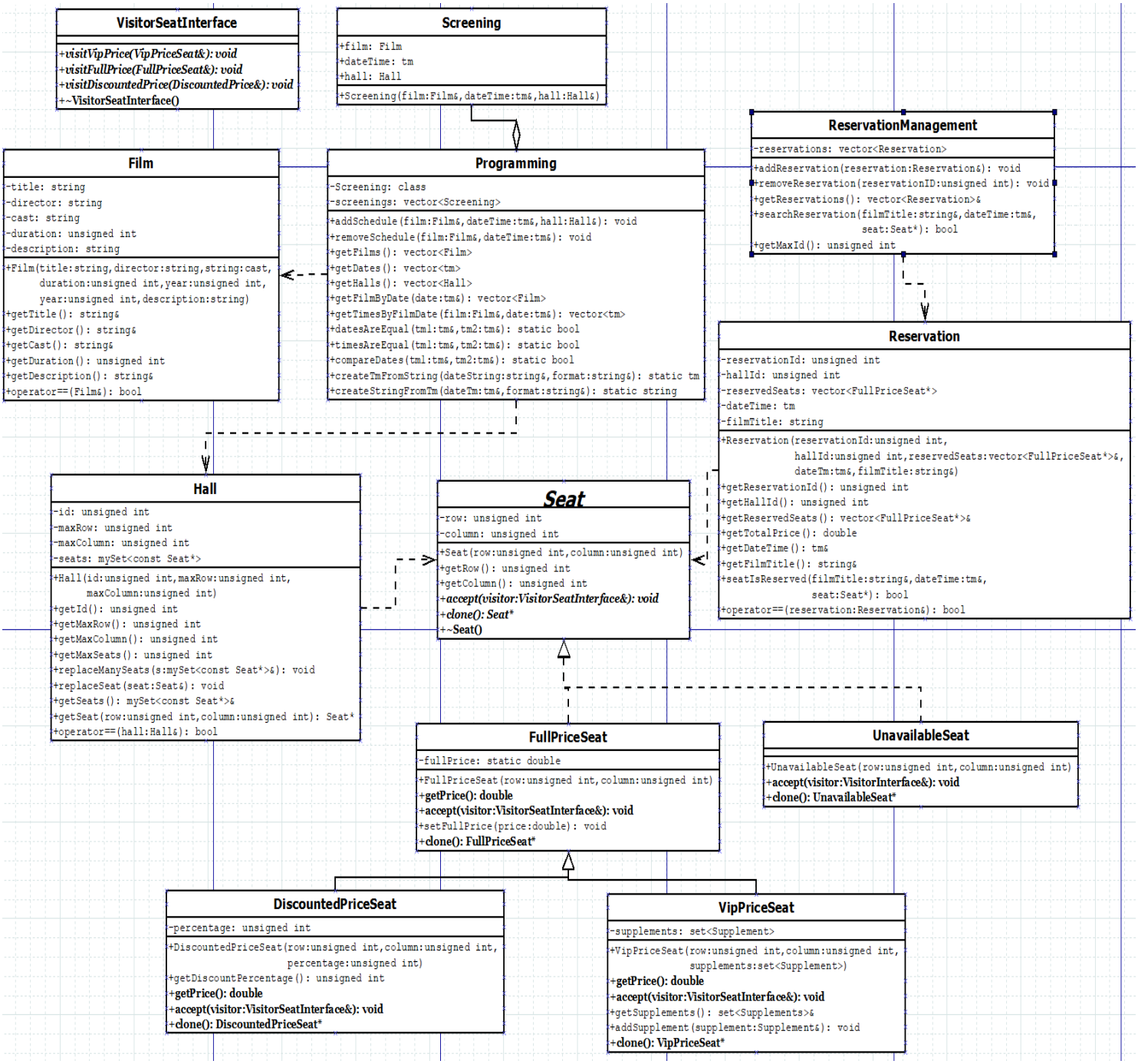
Il modello logico di *CineMatrix* si articola in diverse classi che gestiscono i due principali aspetti del sistema: la programmazione cinematografica e la prenotazione dei posti. Nel diagramma delle classi sono rappresentate le classi principali e le loro relazioni.

Alla base di tutto ciò vi è la classe base astratta ***Seat*** che rappresenta le informazioni di un posto all'interno di una sala cinematografica. Oltre agli attributi riga e colonna, che identificano univocamente il posto nella sala, sono implementati i metodi *getter* per accedere a tali valori. Oltre a ciò, vengono dichiarati due metodi virtuali puri *accept* e *clone.*

Le classi concrete ***ViPPriceSeat, FullPriceSeat, DiscountedPriceSeat***, ***UnavailableSeat*** ereditano dalla classe *Seat* e rappresentano i diversi tipi di posti nella sala cinematografica. In particolare, come mostrato in figura, le classi *DiscountedPriceSeat* e *VipPriceSeat* ereditano da *FullPriceSeat* che a sua volta eredita da *Seat*. Quest'ultime tre classi concrete rappresentano i posti disponibili.

La classe *FullPriceSeat* offre un'esperienza standard a un prezzo medio, mentre la classe *VipPriceSeat* offre un'esperienza di lusso con un prezzo più alto rispetto agli altri tipi di posti in base ai vantaggi disponibili. I supplementi inseriti riguardano per esempio le poltrone reclinabili, il cibo, le bevande e il lettino poggiapiedi. La classe *DiscountedPriceSeat* offre la possibilità di ottenere un posto a un prezzo scontato, il quale dipende dalla percentuale di sconto applicata. Ognuna di queste classi concrete effettua l'implementazione del metodo virtuale *getPrice,* che restituisce il prezzo specifico del posto. Ho scelto questo tipo di architettura perché il posto a prezzo scontato e il posto Vip sono delle specializzazioni del posto a prezzo intero e, in particolar modo, calcolano il prezzo attraverso quest'ultimo in modi diversi.

In quest'ultime tre classi sono disponibili, oltre ai metodi *getter*, anche dei metodi non costanti che permettono di modificare il prezzo intero per ogni istanza di questo tipo di posti, di modificare la percentuale di sconto ma anche di aggiungere benefici nel caso dei posti Vip.



La classe *UnavailableSeat* rappresenta un posto di una sala cinematografica non disponibile e, di conseguenza, non selezionabile e senza prezzo.

Tutte e quattro le classi concrete, quindi, implementano i metodi puri *accept* e *clone.* Nel primo caso si sta utilizzando il design pattern *Visitor*, per la visualizzazione dei posti nella sala cinematografica con colori diversi e per la visualizzazione della loro descrizione. In base al tipo di posto, viene applicato un colore specifico durante la loro selezione e una volta selezionati (tranne nel caso di *Unavailable)* viene fornita una descrizione accurata in base alla scelta. Nell'esempio fornito, i posti VIP sono rappresentati con un bordo rosso, i posti scontati con un bordo verde, i posti a prezzo intero con un bordo blu e infine i posti non disponibili con contorno e sfondo grigio.

L'utilizzo del design pattern *Visitor* permette di separare la logica di visualizzazione dei posti dalla loro implementazione. In questo modo, è possibile estendere le funzionalità di visualizzazione senza modificare la struttura delle classi *Seat* e delle sue sottoclassi. Queste classi e il design pattern Visitor consentono di gestire in modo efficace la rappresentazione grafica dei posti nella sala cinematografica di *CineMatrix*, offrendo un'esperienza visiva chiara agli utenti e permettendo loro di identificare rapidamente i diversi tipi di posti disponibili.

Nel secondo caso, quello dell'implementazione di *clone*, lo scopo è quello di ottenere, attraverso il *clone pattern,* una copia dell'oggetto *Seat* preso in esame. Questo sarà utile nel momento in cui vorremmo evitare inconsistenza dei dati nel momento in cui vengono effettuate condivisioni di posti in sale diverse nell'ambito della classe *Hall.*

Per rappresentare la sala cinematografica nel modello logico di CineMatrix, è stata introdotta la classe ***Hall.*** La classe *Hall* è stata progettata con l'obiettivo di gestire le informazioni relative alla sala cinematografica e ai posti al suo interno. È stata definita con gli attributi *id, maxCol, maxRow* e un set di posti (implementato attraverso un set nativo che ho denominato ***mySet***). L'attributo *id* rappresenta l'identificativo univoco della sala. È stato implementato anche l'overloading dell'operatore di uguaglianza per facilitare il confronto tra oggetti di tipo *Hall.* Gli attributi *maxCol* e *maxRow* rappresentano rispettivamente il numero massimo di colonne e righe all'interno della sala. Questi attributi sono utili per stabilire i limiti di quella che ho deciso di definire come la matrice di posti. Il contenitore di posti rappresenta la disposizione fisica dei posti nella sala cinematografica. Durante la creazione di una nuova istanza di *Hall*, il set di posti viene inizializzato con posti di tipo *FullPriceSeat*. Questo comportamento può essere modificato tramite altri metodi disponibili nella classe.

La classe *Hall* mette quindi a disposizione vari metodi per la gestione dei posti. Il metodo *replaceManySeats* consente di inserire un insieme di posti in determinate posizioni all'interno della sala, verificando che i posti da inserire siano compresi all'interno dei limiti della sala e rimpiazzando i posti presenti in precedenza. In modo simile, il metodo *replaceSeat* permette di rimpiazzare un singolo posto nella sala. Questo metodo è particolarmente importante perché si cerca di gestire il problema di condivisione di posti tra più sale attraverso il metodo *clone* e anche di eliminare elementi rimossi dal contenitore. Infine, il metodo *getSeat* consente di ottenere il posto in una posizione specifica all'interno della sala. Questo metodo è utile per determinare il tipo di posto in una determinata posizione e per ottenere ulteriori informazioni sul posto stesso.

La classe *Hall* rappresenta quindi un'entità essenziale nel modello logico di CineMatrix, consentendo la gestione dei posti all'interno della sala cinematografica e fornendo metodi per il rimpiazzo e l'accesso ai posti.

Per rappresentare l'oggetto film nel modello logico di CineMatrix, è stata introdotta la classe ***Film***. La classe *Film* rappresenta un film all'interno del sistema e include attributi come il titolo, il cast, la descrizione e la durata. L'attributo *title* viene utilizzato come identificatore univoco per il film e viene implementato anche l'overloading dell'operatore di uguaglianza per consentire il confronto tra gli oggetti di tipo *Film*. La classe ***Film*** offre i metodi *getter* che consentono di ottenere le informazioni specifiche relative a un film. Queste informazioni sono fondamentali per consentire agli utenti di visualizzare e selezionare i film durante la programmazione e la prenotazione dei posti

Per rappresentare una delle classi centrali del progetto, ovvero ***Programming***, che gestisce la programmazione dei film in un cinema, è stata introdotta la classe annidata *Screening*. L'attributo principale di *Programming* è un vettore di oggetti *Screening*, che rappresenta le diverse programmazioni dei film. La classe *Screening* rappresenta una singola proiezione di un film e contiene gli attributi di data e orario di programmazione, il film associato (di tipo *Film*) e la sala in cui si svolge (di tipo *Hall*). Questa struttura permette di associare un film a una specifica data e ora di proiezione in una sala cinematografica.

La classe *Programming* offre diverse funzionalità. Il metodo *addSchedule* consente di aggiungere una programmazione al vettore di *Screening*, prendendo come parametri l'orario, la sala e il film associato. Inoltre, è stato implementato il metodo *removeSchedule*, che rimuove una programmazione specifica dal vettore di *Screening*. Per fornire accesso alle informazioni sulla programmazione, sono stati implementati metodi *getter.* Ad esempio, il metodo *getFilms* restituisce l'elenco dei film presenti nella programmazione, il metodo *getDates* restituisce le date delle proiezioni disponibili e il metodo *getHalls* restituisce le sale coinvolte nella programmazione. Questi metodi getter consentono agli utenti di ottenere informazioni sui film, le date e le sale disponibili per la programmazione.

Al fine di fornire funzionalità aggiuntive per gli utenti, sono stati implementati ulteriori metodi getter come *getHall*, che restituisce la sala in cui viene proiettato un determinato film in una specifica data, *getFilmsByDate,* che restituisce i film programmati in una specifica data senza considerare l'ora, e *getTimesByFilmDate*, che restituisce gli orari disponibili di proiezione per un determinato film in una specifica data.
La classe ***Reservation*** rappresenta una singola prenotazione di una programmazione cinematografica. Ogni prenotazione è identificata da un ID univoco e, per questo motivo, viene implementato anche l'overloading dell'operatore di uguaglianza per consentire il confronto tra le prenotazioni. Gli attributi principali di *Reservation* includono *hallId* (identificativo della sala), i posti prenotati, l'orario e la data dello spettacolo e il titolo del film scelto. La classe *Reservation* offre metodi getter per accedere ai valori di questi attributi, consentendo di ottenere informazioni

specifiche sulla prenotazione. Inoltre, è stato implementato il metodo *getTotalPrice*, che sfrutta il polimorfismo della classe *Seat*, per calcolare il prezzo totale della prenotazione. Questo metodo utilizza i posti prenotati per determinare il tipo di posto (ad esempio, VIP, prezzo intero, scontato) e calcola il prezzo totale in base a questi tipi di posti. Per verificare se un posto in una determinata data e ora per un film è occupato o meno, è stato implementato un metodo che utilizza le informazioni di data, ora e posti prenotati. Questo metodo verifica se c'è una sovrapposizione tra la prenotazione corrente e altre prenotazioni già effettuate per il medesimo posto, data e ora.

La classe *Reservation* rappresenta quindi una prenotazione specifica di una programmazione cinematografica, consentendo di accedere alle informazioni sulla prenotazione, calcolare il prezzo totale e verificare la disponibilità di un posto in una data e ora specifiche per un determinato film.

La classe ***ReservationManagement*** rappresenta la gestione delle prenotazioni effettuate nel sistema. Come attributo principale, contiene un vettore di prenotazioni che memorizza tutte le prenotazioni effettuate. La classe *ReservationManagement* fornisce i metodi *addReservation* per aggiungere una prenotazione al vettore e *removeReservation* per rimuovere una prenotazione specifica dal vettore. Inoltre, mette a disposizione un metodo *getter* per ottenere il vettore completo di prenotazioni. In aggiunta alle funzionalità di base, sono state implementate due funzioni aggiuntive. Il metodo *searchReservation* prende in input una programmazione (schedule) e un posto, e cerca se quel posto è già occupato in quella specifica programmazione. Questo metodo consente di verificare se un posto è stato già prenotato per una determinata programmazione cinematografica.

Il metodo *getMaxId* consente di ottenere l'ID massimo utilizzato per le prenotazioni effettuate, che può essere utile per generare nuovi ID univoci per le prenotazioni.

La classe ***ReservationManagement*** svolge quindi un ruolo centrale nella gestione delle prenotazioni effettuate, fornendo funzionalità per l'aggiunta, la rimozione e la ricerca delle prenotazioni.

Infine, è presente l'interfaccia ***VisitorSeatInterface***, che verrà implementata nella classe *VisitorSeatView* utilizzando il design pattern *Visitor.* Questa interfaccia definisce i metodi che il *Visitor* utilizzerà per visitare e manipolare gli oggetti di tipo *Seat*.

Polimorfismo

Nel contesto di *CineMatrix*, la classe *Seat* rappresenta un posto in una sala cinematografica e presenta diverse specializzazioni, come *ViPPriceSeat, FullPriceSeat, DiscountedPriceSeat* e *UnavailableSeat* che offrono caratteristiche specifiche per ciascun tipo di posto e che implementano i metodi puri della classe astratta *Seat, accept e clone.* Inoltre, le prime tre classi concrete citate implementano in modo diverso il metodo virtuale *getPrice*. In caso di *FullPriceSeat* verrebbe resituito il prezzo intero del posto, in caso di *DiscountedPriceSeat* verrebbe restituito il prezzo intero del posto a cui viene applicata una percentuale di sconto e in caso di *VipPriceSeat* verrebbe restituito il prezzo intero del posto a cui vengono applicati dei supplementi di prezzo in base ai vantaggi disponibili. Questo metodo viene utilizzato dalla classe *Reservation* per calcolare il prezzo totale della prenotazione che viene visualizzato sia nel momento della prenotazione che nella lista delle prenotazioni.

Per quanto riguarda il metodo *clone* esso viene utilizzato per creare una copia dell'oggetto della gerarchia di riferimento. In particolare, viene utilizzato nella classe *Hall* nel momento in cui si vede eliminare un oggetto di tipo *Seat* dal contenitore *mySet* consentendo di gestire l'eventuale condivisione di memoria di due posti in più sale diverse.

Tuttavia, l'utilizzo principale del polimorfismo riguarda il *design pattern Visitor* nella gerarchia *Seat.* È presente la classe base astratta *VisitorSeatInterface* che definisce i metodi *visit* per visitare e manipolare i posti e nella classe *VisitorSeatView,* che implementa l'interfaccia *VisitorSeatInterface*, sono definiti i metodi *visit* per le diverse specializzazioni dei posti.
Il *design pattern Visitor* viene utilizzato per ricavare il prezzo di ogni posto e per la costruzione dei *widget* per mostrare le diverse tipologie di elementi oltre che per costruire descrizioni che rispettino gli attributi dei posti. Rispetto a quest'ultimo punto, la visualizzazione prevede tre possibili rese grafiche.

- *FullPriceSeat:* è visualizzato tramite un oggetto *QCheckBox* il cui contorno è blu e se cliccato assume uno sfondo blue
- *VipPriceSeat:* è visualizzato tramite un oggetto *QCheckBox* il cui contorno è rosso e se cliccato assume uno sfondo rosso

- *DiscountedPriceSeat*: è visualizzato tramite un oggetto *QCheckBox* il cui contorno è verde e se cliccato assume uno sfondo verde
- *UnavailableSeat*: è visualizzato tramite un oggetto *QCheckBox* disabilitato il cui contorno è grigio con sfondo grigio.

Questa distinzione grafica permette agli utenti di identificare facilmente i posti con caratteristiche diverse e di effettuare le scelte di prenotazione in modo intuitivo.

Inoltre, ogni posto possiede una descrizione specifica che viene visualizzata sia tramite *TicketInfoWidget* che tramite *Reservation* List attraverso il widget SeatsInfoWidget utilizzando allo stesso modo il *design pattern Visitor* e sfruttando le diverse proprietà sopra elencate.

- *FullPriceSeat:* viene visualizzata una semplice descrizione del posto
- *VipPriceSeat:* oltre alla descrizione del posto vengono elencati i vari supplementi di prezzo e la loro motivazione (attraverso un elenco di benefici) oltre che il prezzo intero di partenza.
- *DiscountedPriceSeat*: oltre alla descrizione del posto viene esposto il prezzo intero con la relativa percentuale di sconto
- *UnavailableSeat*: non è presente una descrizione in quanto non selezionabile.

Attraverso il *design pattern Visitor* e il polimorfismo è possibile dunque trattare oggetti di tipo *Seat* in maniera generica, consentendo di gestire posti con caratteristiche diverse in modo uniforme.

Persistenza dei dati

La persistenza dei dati viene gestita tramite l'utilizzo di file di testo e la libreria standard *fstream* disponibile di *C++*. Il formato del file utilizzato è un formato di dati delimitato da un carattere '|'. Ogni riga del file rappresenta una prenotazione e contiene una serie di campi separati dal carattere delimitatore.

I campi presenti in ogni riga includono:

- reservationId: l'identificatore univoco della prenotazione
- totalPrice: il prezzo totale della prenotazione
- filmTitle: il titolo del film prenotato
- hallId: l'identificatore della sala cinematografica
- dateString: una stringa che rappresenta la data e l'ora della proiezione

Dopo questi campi, seguono una serie di coppie di valori che rappresentano le coordinate delle sedute prenotate. Ogni coppia è composta da rowString (rappresentante la riga della seduta) e columnString (rappresentante la colonna della seduta). Queste coppie di valori sono separate dal carattere delimitatore '|'.

Per quanto riguarda la struttura dei dati nel file, ogni riga corrisponde a una prenotazione e contiene tutte le informazioni necessarie per ricostruire l'oggetto di tipo *Reservation*. I campi sono separati dal carattere delimitatore '|' per consentire la divisione dei dati durante la lettura del file.

Quando l'applicazione di CineMatrix viene aperta, viene creato un oggetto di tipo *ReservationList* che si occupa di visualizzare le informazioni sulle prenotazioni effettuate. Ogni volta che l'utente clicca sul pulsante "*Reservations*", viene eseguito un refresh per verificare se ci sono state nuove scritture sul file e creare una nuova lista di prenotazioni aggiornata.

L'aggiornamento della lista delle prenotazioni avviene in tempo reale, in quanto viene riletto il contenuto del nuovo file di testo che contiene le informazioni sulle prenotazioni. In questo modo, se è stata eliminata una prenotazione tramite il pulsante "*Delete*", la lista delle prenotazioni viene immediatamente aggiornata, rimuovendo la prenotazione eliminata dalla visualizzazione.

Questa funzionalità di aggiornamento in tempo reale permette agli utenti di avere sempre accesso alle informazioni più recenti sulle prenotazioni effettuate, garantendo una corretta visualizzazione delle prenotazioni presenti nel sistema.

Un esempio della struttura dei file è dato dall'esempio è rappresentato dal file di testo "*reservation.txt*".

Funzionalità implementate

Le funzionalità implementate sono suddivise in due categorie principali: la gestione della lista dei film e la gestione della lista delle prenotazioni.

Per quanto riguarda la lista dei film, è presente un *QTabWidget* che consente di selezionare una data specifica per visualizzare gli spettacoli disponibili. Ogni tab rappresenta una data e contiene informazioni come il titolo del film, il cast, la durata e la sinossi. Inoltre, sono elencati gli orari di programmazione cliccabili per ogni film. Se si clicca su un orario, viene aperta una finestra che mostra la disposizione dei posti nella sala e la loro disponibilità. I posti con contorno ma vuoti sono posti liberi e selezionabili, mentre i posti con contorno ma pieni e colorati di nero indicano che sono già prenotati e quindi non selezionabili come i posti con sfondo e contorno grigio che rappresentano la classe *Unavailable*. Vi è un tasto di conferma che permette di confermare i posti selezionati.

Nel caso in cui si tenti di confermare senza aver selezionato alcun posto, viene mostrata una finestra di avviso che segnala l'impossibilità di procedere senza aver selezionato almeno un posto. Se invece si conferma la selezione, viene visualizzata una schermata riassuntiva con i dettagli della prenotazione, tra cui il titolo del film, la data e l'ora dello spettacolo e per ogni posto selezionato il relativo prezzo e la posizione con una descrizione dettagliata. Viene inoltre mostrato il prezzo totale della prenotazione. Al termine, la prenotazione viene confermata e i posti selezionati vengono considerati prenotati.

Per visualizzare la lista delle prenotazioni effettuate, è presente un pulsante "*Reservations*" che apre una finestra contenente la lista delle prenotazioni. Ogni prenotazione mostra il titolo del film, la data e l'ora dello spettacolo, la sala e il prezzo totale. Sono disponibili due pulsanti cliccabili: "*Info*" permette di visualizzare le caratteristiche dei posti prenotati, mentre "*Delete*" consente di eliminare una prenotazione. La lista delle prenotazioni si aggiorna in tempo reale, sia sulla schermata che nel file di memorizzazione, dopo l'eliminazione di una prenotazione. Al passaggio del mouse sopra i pulsanti cliccabili e sottolineati il cursore assume l'icona appropriata per evidenziare l'interattività.
Per tornare alla visualizzazione della lista dei film, è possibile cliccare sia sul tasto "*Home*" che sul tasto "*CineMatrix*" che si trova in alto nella schermata principale. Entrambi i pulsanti consentono di tornare alla schermata iniziale in cui vengono mostrati i film disponibili e le relative informazioni.
Alcune accortezze aggiuntive sono state:

- "Tagliare" il titolo del film se troppo lungo nella lista delle prenotazioni
- Utilizzare *QScrollarea* per la visualizzazione delle finestre di dialogo *TicketInfoWidget* e *SeatsInfoWidget*
- Assicurarsi che i posti prenoti non siano più selezionabili (attraverso la funzione *setDisabled*)
- Visualizzare in modo diverso i titoli dei film o le informazioni in risalto tramite il grassetto o sottolineando alcune *QLabel*
- L'utilizzo di colori e di stili grafici

Le funzionalità elencate sono intese in aggiunta a quanto richiesto dalle specifiche del progetto.

Rendicontazione ore

Attività	Ore Previste	Ore Effettive
Studio e progettazione	10	6
Sviluppo del codice del modello	10	14
Studio del framework Qt	10	10
Sviluppo del codice della GUI	10	18
Test e debug	5	10
Stesura della relazione	5	5
totale	50	**63**

Il monte ore previsto per il progetto è stato superato a causa di alcune sfide impreviste durante lo sviluppo. Durante la fase di sviluppo del codice del modello, sono state necessarie scelte architetturali complesse che hanno richiesto più tempo del previsto. Inoltre, lo studio del framework Qt è stato particolarmente difficile a causa del numero e della complessità dei widget implementati e della necessità di mantenere la coerenza della logica all'interno dell'applicazione oltre al fatto che l'utilizzo di un framework mi era nuovo.

Durante il processo di sviluppo, sono stati riscontrati diversi bug e warning che richiedevano tempo e sforzi per essere interpretati e risolti correttamente. Inoltre, l'utilizzo della struct tm per memorizzare una data non è stato immediato da comprendere e implementare correttamente, ciò ha portato alla scelta di realizzare delle funzioni apposite situate in *Programming* che ne facilitassero l'uso.

A causa del superamento del monte ore previsto, non è stato possibile implementare ulteriori funzionalità volte all'estetica del progetto. Queste idee includevano la visualizzazione delle informazioni dettagliate di un film, inclusi gli orari delle proiezioni, una barra di ricerca per facilitare la ricerca di film specifici e una resa visiva più accattivante.

Nonostante queste difficoltà, il progetto è stato portato a termine, concentrando gli sforzi sulle funzionalità principali e sulla risoluzione dei bug critici.