RESIDENCE STELLA

Progetto del corso Programmazione ad Oggetti
Università di Padova - anno 2020-2021
Relazione di Benetazzo Irene 1223865
Progetto sviluppato in coppia con Lazzarin Nicola 1204686

INDICE

INTRODUZIONE

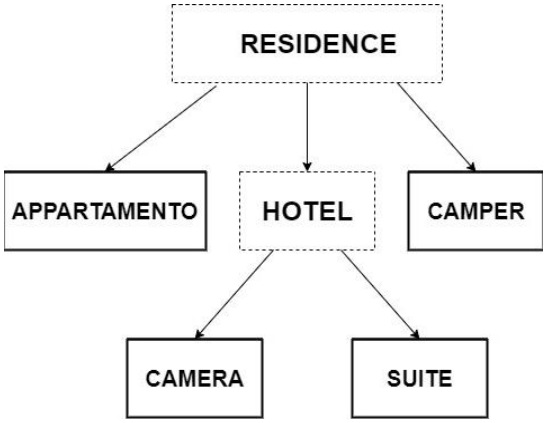
Residence Stella è un’applicazione pensata per presentare ai clienti l’offerta delle strutture disponibili per passare un piacevole e rilassante soggiorno al mare.
Le strutture proposte si suddividono in 4 categorie permettendo un’ampia scelta: camera d’albergo, suite, appartamento, piazzola camper; per ogni tipologia si hanno varie dimensioni e prezzi. Oltre ai specifici servizi proposti in base alla tipologia di struttura è disponibile il servizio di accesso alla piscina e il servizio di prenotazione di un posto riservato in spiaggia.
Lo scopo dell’applicazione è mostrare le varie strutture offerte dal residence Stella e il prezzo a notte considerando il numero di ospiti effettivo e i vari servizi richiesti.
La prenotazione si effettua esclusivamente tramite telefono chiamando direttamente il residence Stella.
Per sviluppare il progetto è stato usato il design pattern Model-Controller-View

MODELLO

Il modello del progetto è costituito dalla gerarchia 'Residence', dal contenitore che è un vettore templetizzato, ed è presente un template di puntatori smart che gestisce la memoria in maniera profonda. Il modello è stato progettato e scritto in modo indipendente rispetto alla GUI e si potrà utilizzare anche in futuro.

GERARCHIA

La gerarchia ha altezza due ed è costituita da 6 classi di cui 2 astratte e 4 concrete.



RESIDENCE

É la classe base astratta della gerarchia che raggruppa tutte le possibili strutture che possono essere offerte al cliente. Sono presenti vari campi dati privati che rappresentano le principali e basilari caratteristiche che presentano tutte le strutture.
Oltre ai get e set dei campi dati privati e al distruttore virtuale, ci sono 4 funzioni virtuali pure che verranno implementate successivamente, un'altra funzione virtuale e una non virtuale.
- *virtual Residence* clone() const=0* permette la gestione della memoria profonda in deep_unique_ptr e questa funzione verrà implementata solo nelle classi concrete per permettere la clonazione degli oggetti.

- *virtual double costoPiscina()=0* calcola il costo della piscina.
- *virtual double costoSpiaggia()=0* calcola il costo della spiaggia.
- *virtual double prezzo()=0* calcola il prezzo totale di una notte in una determinata struttura con n partecipanti e i servizi richiesti.
- *virtual aumentoStagione(periodo p)* aumenta in percentuale il prezzo in base al periodo (è stato definito un tipo enum per indicare la stagione: bassa, media, alta)
- *double prezzoMinStagione()* funzione non virtual che calcola il prezzo minimo della struttura data la stagione

HOTEL

É un'altra classe astratta della gerarchia che deriva da residence, e implementa una nuova funzione *virtual double costoRistorante()* e calcola il costo del ristorante in base al numero di ospiti e alla scelta fatta (obbligatoria almeno la colazione).
Viene fatto l'override di due funzioni virtuali pure di residence:
- *virtual double costoPiscina() override* la piscina è gratuita per i clienti dell'hotel
- *virtual double costoSpiaggia() override* il posto riservato in spiaggia per l'hotel si paga a camera.

La classe hotel è astratta perchè non è stato fatto l'override di tutte le funzioni virtuali pure di residence, infatti prezzo() e clone() sono ancora virtuali pure.

CAMERA

É una classe concreta che deriva da hotel (e a sua volta da residence), sono state implementate le due funzioni virtuali pure (prezzo e clone). Il prezzo della camera per una notte viene calcolato a partire dal prezzo minimo e l'aumento in base alla stagione, il ristorante e se richiesti il servizio piscina e/o spiaggia (anche se per i clienti dell'hotel la piscina è gratuita, nel calcolo del prezzo è comunque invocata la funzione perchè in futuro il codice si può modificare rendendo la piscina a pagamento).

SUITE

É una classe concreta che deriva da hotel (e a sua volta da residence), la suite può essere costituita da più stanze, inoltre gli ospiti possono richiedere un ufficio privato e/o un centro benessere. E' stato fatto l'override di *costoRistorante* in quanto c'è la possibilità di richiedere il servizio in camera a pagamento, override *aumentoStagione* perchè è stata alzata la percentuale e infine override della funzione prezzo: al prezzo minimo si aggiunge il costo dell'eventuale ufficio e/o centro benessere richiesto, aumento percentuale in base alla stagione, ristorante, servizio piscina e/o spiaggia se richiesti.

APPARTAMENTO

É una classe concreta che deriva da residence, in base al numero delle stanze l'appartamento può essere monolocale, bilocale, trilocale. I fornelli presenti possono essere a gas o ad induzione, inoltre in alcuni appartamenti è presente la lavatrice. E' obbligatorio pagare la pulizia dell'appartamento mentre se il cliente lascia la cucina perfettamente pulita e in ordine non è necessario pagare il supplemento pulizia cucina, il costo della pulizia dell'appartamento è calcolato da una funzione virtuale (la funzione è virtuale in ottica futura). Inoltre è presente l'override di costo piscina e costo spiaggia che si pagano a persona con i prezzi standard del residence; infine l'override di prezzo: oltre al prezzo minimo si paga un fisso a persona, le pulizie, aumento percentuale in base alla stagione e servizio piscina e/o spiaggia se richiesti.

CAMPER

É una classe concreta che deriva da residence, cioè si offre una zona di terra con la presenza di una colonnina di corrente e accesso all'acqua (se richiesti dal cliente); alcune delle informazioni base presenti nella classe astratta residence qui sono posti a zero di default come il piano, disabile, numero letti disponibili in quanto non sono necessarie in una piazzola camper. Le funzioni per calcolare il costo dell'acqua e della corrente sono virtuali e inoltre è stato fatto l'override di costo piscina e costo spiaggia che si pagano a persona con i prezzi standard del residence; infine è presente l'override di prezzo: oltre al prezzo minimo si paga un fisso a ospite, aumento percentuale in base alla stagione e i servizi richiesti (acqua, corrente, piscina, spiaggia).

POLIMORFISMO

Il polimorfismo è stato ampiamente usato nella gerarchia:
-distruttore virtuale: distrugge gli oggetti
-clone: clona gli oggetti e rende possibile la gestione della memoria profonda di deep_unique_ptr
-aumentoStagione: aumento percentuale del prezzo in base alla stagione
-costoPiscina: costo dell'accesso alla piscina
-costoSpiaggia: costo servizio spiaggia
-costoRistorante: (in hotel e le sue classi derivate) calcola il costo del ristorante in base alla pensione scelta: solo colazione, mezza pensione, pensione completa
-puliziaFinale: (in appartamento) calcola il costo della pulizia dell'appartamento e può essere compresa la pulizia della cucina
-costoAcqua: (in camper) calcola il costo dell'acqua giornaliero per persona
-costoCorrente: (in camper) calcola il costo della corrente giornaliero per persona
Di queste ultime tre funzioni virtuali non è stato fatto nessun override nel progetto ma sono comunque virtuali in ottica futura di estensibilità del progetto.

CONTENITORE

Come contenitore è stato scritto vettore che è una versione più breve del contenitore standard vector offerto dalla libreria STL, implementando tutte le funzioni basilari e utili oltre agli iteratori.
E' stato scritto in forma di template e non con il tipo specifico così da permetterne il suo riutilizzo in altri progetti; la class vettore contiene al suo interno due classi: iterator e const_iterator nelle quali sono state implementati i metodi principali e utili degli iteratori.

PUNTATORE POLIMORFO "SMART"

Per implementare un'automatica gestione profonda della memoria è stato definito deep_unique_ptr che è una versione ridotta dello std::uinque_ptr<T>.
Per sfruttare ciò è necessario che il tipo dell'oggetto istanziato supporti la clonazione e la distruzione polimorfa, quindi in gerarchia è necessario il metodo virtuale tipo* clone() const che viene implementato solo nelle classi concrete ritornando un puntatore di quel tipo al nuovo spazio sullo heap.

INPUT

Il file di input dati è un semplice .txt che viene letto dalla classe database.cpp usando alcune classi di Qt: QFile e QTextStream, i dati mentre vengono letti passano al model che crea i rispettivi oggetti.
Nella parte iniziale del file database.txt viene spiegato come scrivere i dati di input per ogni specifica tipologia di struttura.
Il programma per funzionare è dipendente dal file di input in quanto l'applicazione dopo aver fatto selezionare i servizi carica il vettore per la tipologia di struttura scelta e poi ricava da esso le informazioni utili da mostrare nella schermata 'elenco strutture' e poi successivamente nella finestra 'informazioni e servizi struttura'. Nella consegna è stato fornito il file database.txt (contenente dati inventati) per poter testare il programma.

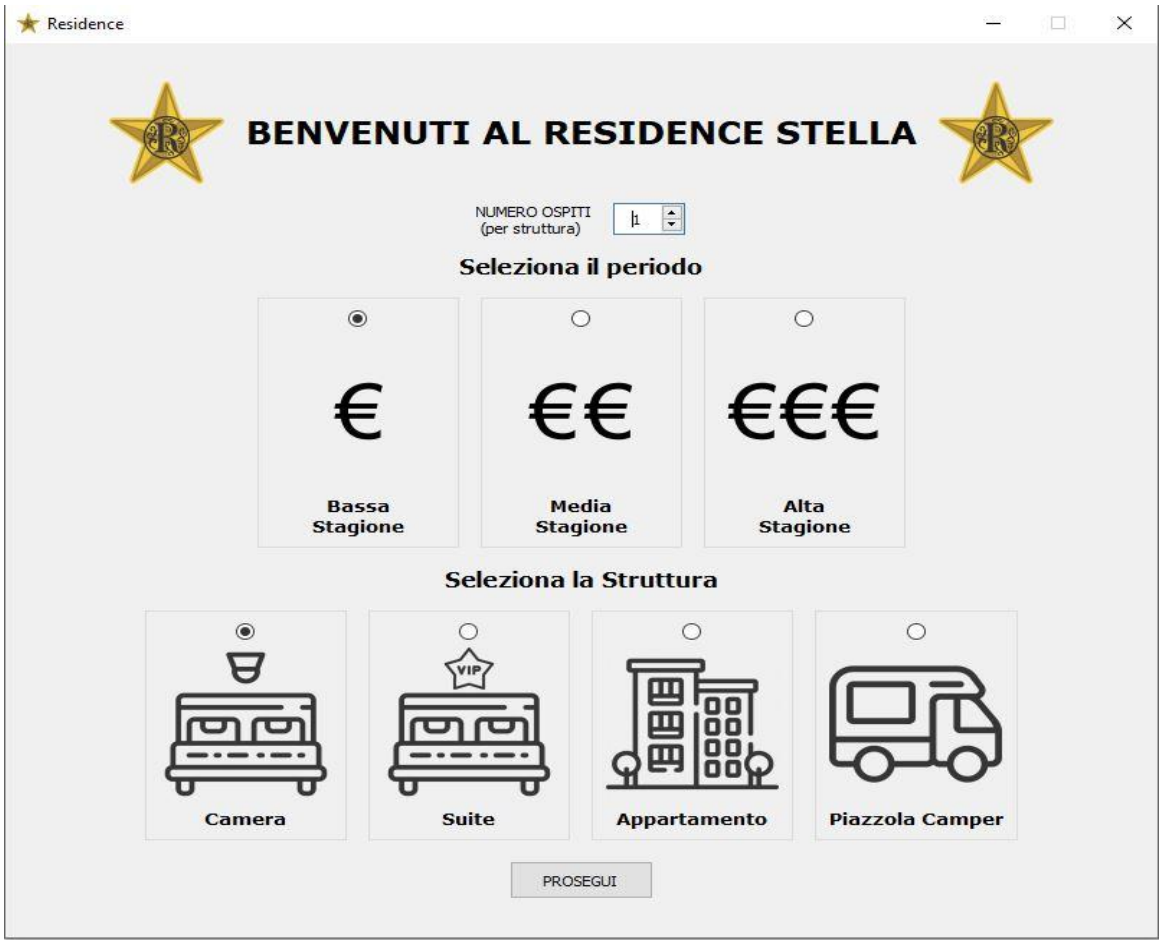
RISORSE

Per far funzionare correttamente il progetto è necessario il database mentre per curare l'estetica del progetto sono necessarie delle immagini. Queste risorse vengono caricate mediante il file QRC (Qt Resource System) direttamente nell'eseguibile.

GUI

La vista è organizzata in 4 schermate e 1 popup informativo (presente nella schermata 3, cioè in 'elenco strutture'). La grandezza delle finestre è fissa e non si può ridimensionare.

HOME



Prima schermata con cui si apre l'applicazione e si deve dire il numero di ospiti (massimo 8), scegliere il periodo e la tipologia di struttura in cui si desidera soggiornare (questa scelta condiziona le successive schermate).
Come pulsante di scelta è stato usato QRadioButton per permettere un'unica scelta e sono già state preselezionate bassa stagione e camera perchè non è possibile proseguire senza selezionare nulla.
La selezione può avvenire tramite click sul pulsante o si può anche cliccare sull'immagine.
Infine è presente il pulsante prosegui da cliccare per andare alla vista successiva

SERVIZI

Questa schermata è condizionata dalla tipologia di struttura scelta in home in quanto sono visualizzati i servizi inerenti:

Camera d'albergo

Si può scegliere se avere il posto riservato in spiaggia, mentre la piscina è già selezionata di default essendo gratuita per i clienti dell'hotel; essendo servizi a scelta è stato utilizzato un QCheckBox. Mentre è necessario scegliere come si vuole usufruire del ristorante ed è obbligatorio fare almeno la colazione, se si seleziona mezza pensione significa che si desidera usufruire della colazione e cena al ristorante, mentre pensione completa comprende colazione, pranzo e cena.

Suite

Si può scegliere se avere il posto riservato in spiaggia, mentre la piscina è già selezionata di default essendo gratuita per i clienti dell'hotel. Mentre è necessario scegliere come si vuole usufruire del ristorante ed è obbligatorio fare almeno la colazione, se si seleziona mezza pensione significa che si desidera usufruire della colazione e cena al ristorante, mentre pensione completa comprende colazione, pranzo e cena; si può scegliere se ricevere il servizio ristorante in camera (è anche già preselezionato, ma si può togliere la spunta per non pagare il servizio). Inoltre si può richiedere di riservare il centro benessere e/o di avere a disposizione un ufficio/sala riunioni privato.

Appartamento

Si può scegliere se avere il posto riservato in spiaggia e/o l'accesso in piscina, inoltre è obbligatorio specificare se la pulizia della cucina deve essere effettuata dal residence e quindi comporta un aumento sul costo della pulizia dell'appartamento.

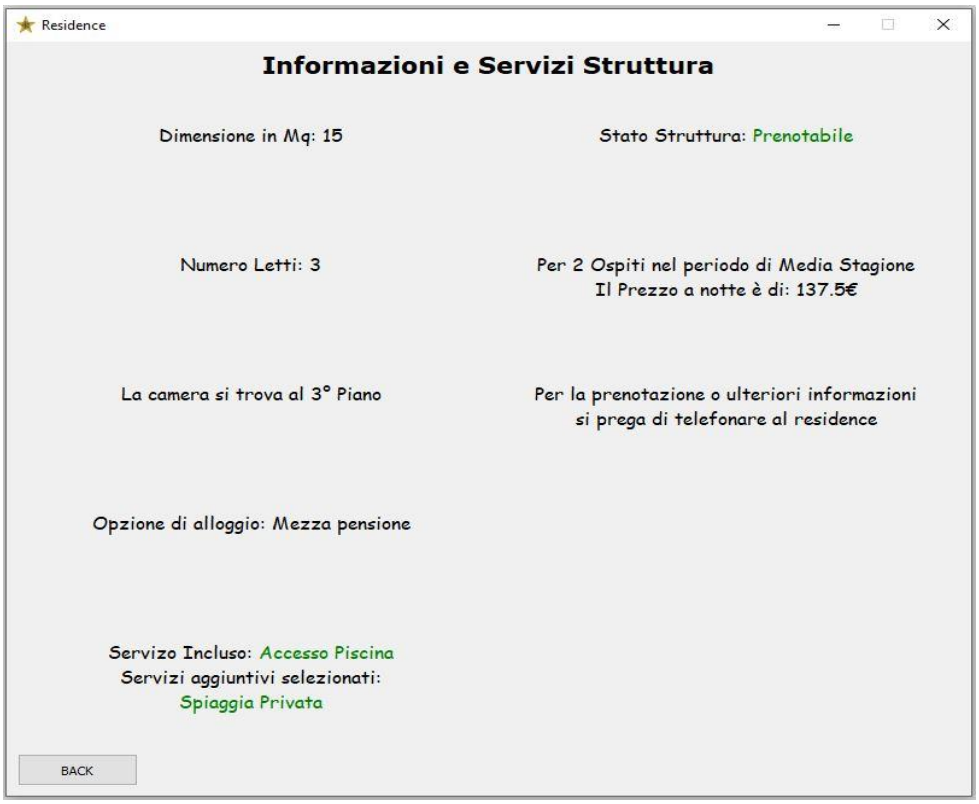
Piazzola camper

Si può scegliere se avere il posto riservato in spiaggia e/o l'accesso in piscina, ed è possibile richiedere di attaccarsi alla colonnina di corrente elettrica 220V e/o di potersi attaccare alla pompa dell'acqua.
Infine in basso è presente il pulsante home per tornare in home e il pulsante prosegui per vedere la successiva schermata, a questo punto avviene il caricamento del vettore dal file database ma solo delle strutture della tipologia selezionata così da evitare inutili caricamenti.

ELENCO STRUTTURE

Questa schermata è costituita da una tabella che visualizza tutto l'elenco delle strutture offerte di quella determinata tipologia di struttura con le principali informazioni utili della struttura. Sopra la tabella è presente un ? che se cliccato si apre un popup che spiega il significato dei vari simboli presenti nella tabella, si noti che se compare il simbolo del letto con una croce sopra significa che quella specifica struttura non ha un numero di letti sufficiente per tutti gli ospiti ma è comunque disponibile se eventualmente si decide di suddividere gli ospiti in più strutture. L'organizzazione delle colonne della tabella varia in base alla tipologia scelta comunque è sempre visibile il prezzo minimo della struttura considerando già l'eventuale aumento percentuale dato dalla stagione, quando la grandezza della finestra non basta per visualizzare tutto l'elenco, appare in automatico una barra di scorrimento.
Sotto la tabella è presente il pulsante back per tornare alla schermata dei servizi e home per tornare direttamente in home, non è presente un pulsante prosegui perchè l'applicazione è terminata ma è solo possibile visualizzare ulteriori informazioni e dettagli di una specifica struttura cliccando due volte sulla rispettiva riga.

INFORMAZIONI E SERVIZI STRUTTURA



Questa schermata visualizza le informazioni complete e più dettagliate di una specifica struttura, vengono specificati anche i vari servizi selezionati in precedenza. Nella colonna di destra si trova riassunto lo stato della struttura e il prezzo complessivo di una notte in quella struttura, poi come riportato si invita a telefonare al residence; l'organizzazione e la spaziatura delle informazioni varia leggermente in base alla tipologia della struttura.
In questa schermata in basso è presente un unico pulsante back che permette di tornare alla schermata di elenco delle strutture.

Per chiudere l'applicazione è sufficiente cliccare sulla classica X in alto a destra.

ORGANIZZAZIONE DEL LAVORO

Io personalmente mi sono occupata del modello mentre il mio compagno Lazzarin Nicola della GUI

SUDDIVISIONE DEI FILE

BENETAZZO IRENE 1223865
vettore.h
deep_unique_ptr
residence.h
residence.cpp
hotel.h
hotel.cpp
camera.h
camera.cpp
suite.h
suite.cpp
appartamento.h
appartamento.cpp
camper.h
camper.cpp
model.h
model.cpp

LAZZARIN NICOLA 1204686
clickablelabel.h
clickablelabel.cpp
home.h
home.cpp
infodialog.h
infodialog.cpp
servizi.h
servizi.cpp
structureview.h
structureview.cpp
structuredetailsview.h
structuredetailsview.cpp
view.h
view.cpp
controller.h
controller.cpp
database.cpp

ORE IMPIEGATE

Visione dei video di tutorato 8h
Analisi del problema, progettazione del progetto e suddivisione dei compiti 5h
Scrittura del vettore e puntatore profondo 15h
Scrittura della gerarchia e file model 15h
Bozza della view 2h
Test progetto 2h
Scrittura relazione 3h
Totale 50 ore.

AMBIENTE DI SVILUPPO

Il progetto è stato sviluppato e testato in un pc Windows 10 con qt 5.12.2 e compilatore MinGW 8.1.0 64bit.
Inoltre è stato testato sulla macchina virtuale Linux-Ubuntu con compilatore GNU g++ 7.3 e qt 5.9.5

ISTRUZIONI PER LA COMPILAZIONE

Per compilare il progetto nella macchina virtuale Ubuntu fornita dal professore è sufficiente:
estrarre la cartella Residence dallo zip
Aprire il terminale e andare sulla cartella unzippata ed eseguire i seguenti comandi:
mkdir build
cd build
qmake ../Residence.pro
make *(aspettare che crei tutti i codici oggetto dei vari file)*
./Residence

Nota: Nella consegna è incluso un file di esempio di input e utile a testare il programma, senza file non è possibile far funzionare l'applicazione.