Relazione sul progetto del corso di Programmazione e Modellazione a Oggetti

Sessione Invernale 2021/2022

Studentessa:
Dalia Abbruciati - matricola 277914

1. Specifica del software

Si intende sviluppare un programma che simuli un'applicazione per la gestione delle prenotazioni di un ristorante dal punto di vista del ristoratore. Nello specifico, nella schermata iniziale si potranno scegliere varie opzioni come registrare la prenotazione di un tavolo oppure ordinare cibi d'asporto o con consegna a domicilio. Il ristoratore potrà inoltre visualizzare una lista con le informazioni sui clienti e su gli ordini effettuati da essi, sia per coloro che stanno mangiando al ristorante sia per coloro che hanno scelto l'asporto/domicilio.

Se un cliente chiede di ordinare l'asporto/domicilio, verrà visualizzato il menù del ristorante in cui il ristoratore selezionerà i cibi richiesti dal cliente e una volta confermato il riepilogo dell'ordine, potrà inserire i dati del cliente come nome, orario in cui desidera ritirare/ricevere il cibo, un recapito telefonico e [per il domicilio] l'indirizzo di consegna.

Per quanto riguarda la gestione dei tavoli invece, quando un cliente chiamerà per prenotarne uno, all'inizio il ristoratore visualizzerà una schermata che mostra la mappa del locale con le varie zone dal quale potrà selezionare quella che il cliente desidera e vedere quanti tavoli sono ancora disponibili per ogni zona. Dopo aver inserito le informazioni utili come nome del cliente, numero di persone per tavolo, recapito telefonico, ora di arrivo, data (nel caso volesse prenotare per una data in particolare) e note (nel caso ci siano richieste particolari sul cibo o per determinati eventi) il gestore cliccando su "Scegli menù" verrà reindirizzato alla schermata del menù dal quale potrà selezionare i cibi che il cliente ha già in mente di ordinare oppure semplicemente saltare questo passaggio e confermare la prenotazione. Durante il servizio, cliccando su "Lista tavoli" il gestore potrà vedere la schermata dettagliata di ogni tavolo diviso per zona, il numero univoco e il tempo che ogni tavolo sta attendendo dall'ultimo ordine effettuato. In base a questo i tavoli cambieranno colore da verdi diventeranno arancioni se il tempo supera i 20 minuti e poi rossi se si superano i 40 minuti [per la simulazione i tavoli cambieranno colore ogni 2 minuti]. Cliccando su ogni singolo tavolo si potranno visualizzare le informazioni sul cliente e il riepilogo dell'ordine, eventualmente modificarlo e infine chiuderlo quando si cliccherà sul pulsante "Paga".

2. Studio del problema

I punti critici riscontrati nella progettazione dell'applicazione sono stati:

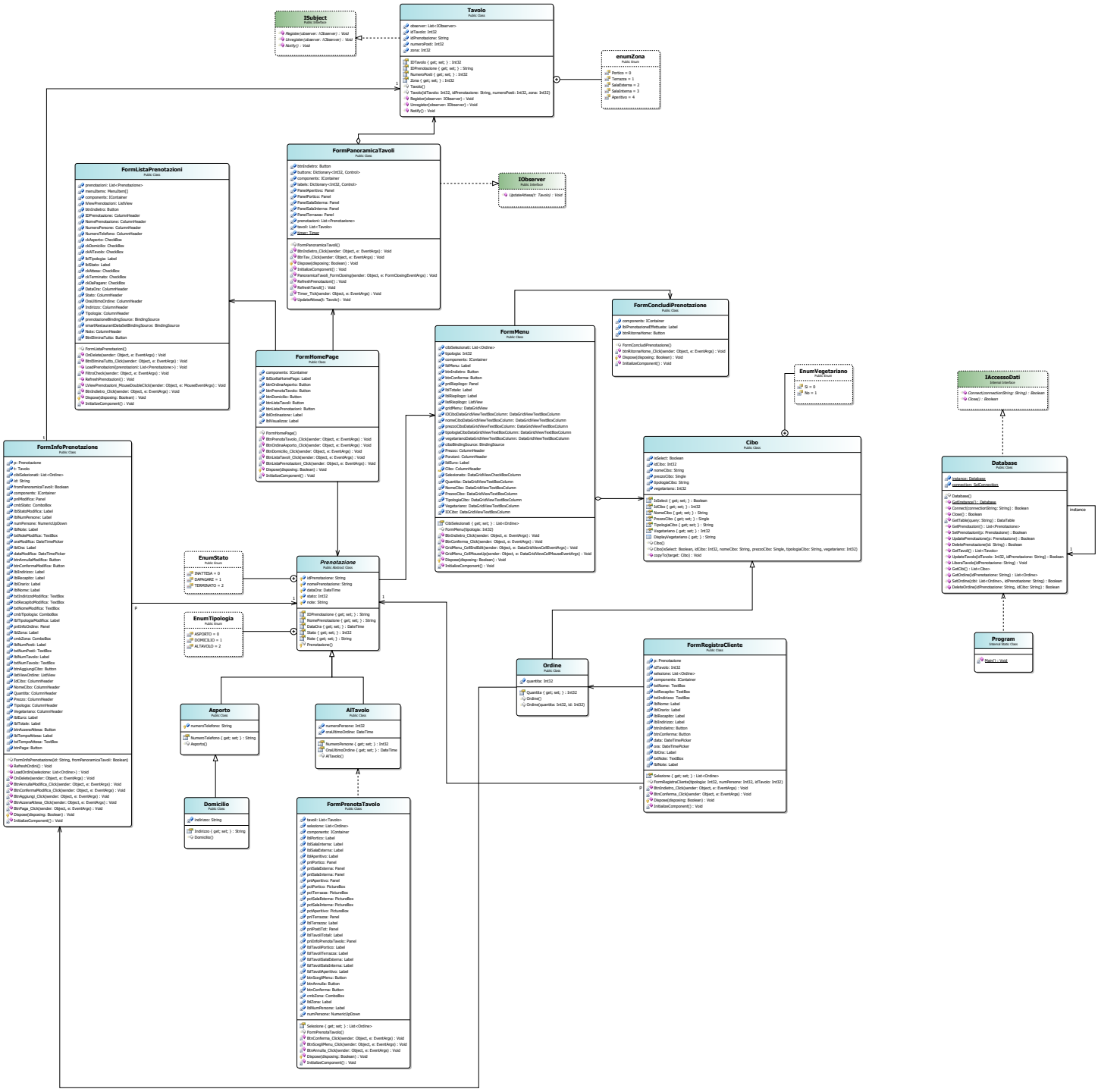
1) Contenere i dati e collegare il database esterno al progetto;
2) Aggiornare il tempo di attesa dei tavoli;

I punti citati precedentemente sono stati risolti nel seguente modo:

1) Per contenere i dati si è scelto di utilizzare il database Microsoft SQL Server. All'interno del progetto ho creato una classe statica '*Database*' che contenesse i vari metodi di connessione e chiusura del DB. Inoltre, per traferire all'interno del programma le varie tabelle ho creato metodi di 'get' e 'set' per ogni tabella e anche metodi di 'update' e 'delete' per aggiornare ed eliminare dati dalle tabelle 'Cibo', "Prenotazione", "Tavolo" e "Ordine".

2) Per quanto riguarda l'aggiornamento del tempo di attesa e del colore del tavolo nell'interfaccia grafica, per centralizzare l'operazione, si è utilizzato il pattern Observer, con la relativa implementazione delle interfacce e dei metodi. Infatti, esso, fa riferimento al metodo "*UpdateAttesa*" della classe "*FormPanoramicaTavoli*" che aggiorna ogni minuto il tempo di attesa del cliente al tavolo a partire dall'ora del suo ultimo ordine effettuato. Inoltre, è stato creato un metodo "*Timer_tick*" che minuto per minuto notifica l'Observer con la lista delle prenotazioni aggiornata.

3. Scelte architetturali



Il diagramma delle classi, sopra rappresentato, raffigura l'architettura principale dell'applicazione. Una volta avviato il programma viene subito istanziata la classe '*Database*', che usa il pattern Singleton e che deriva dall'interfaccia '*IAccessoDati*' che si occupa di eseguire le operazioni di connessione e chiusura al database. La classe

principale, sulla quale si basa quasi interamente l'applicazione, è la classe astratta '*Prenotazione'*. Essa, infatti, gestisce la parte logica e si occupa di raccogliere e mantenere i dati relativi al cliente al momento della prenotazione. Il cliente ha la possibilità di scegliere tra 3 diverse tipologie di prenotazione, *'Asporto'*, *'Domicilio'*, che deriva dalla precedente ma aggiunge il campo contenente l'indirizzo del cliente e '*AlTavolo'* che è collegata alla form *'PrenotaTavolo'* dal quale si potrà scegliere la zona del ristorante desiderata e il numero di persone. Queste ultime 3 classi fanno riferimento alla form '*Menu'* in relazione di aggregazione con i vari cibi che scegliendoli e definendo quindi una quantità si creerà un ordine. Dopo aver registrato le informazioni del cliente, la prenotazione potrà essere revisionata nella schermata '*InfoPrenotazione'*. Dalla form '*HomePage'* è possibile, inoltre, visualizzare la lista completa delle prenotazioni e la '*PanoramicaTavoli'*, che è "figlia" dell'interfaccia '*IObserver*' che servirà per aggiornare il tempo di attesa dei clienti ai tavoli. Essa è inoltre legata alla classe '*Tavolo*', derivata dall'interfaccia '*ISubject*' che si occupa di registrare, disiscrivere e notificare i singoli tavoli al pattern Observer.

I design pattern utilizzati nel progetto sono:

- **MVC**: questo pattern consente di dividere i compiti tra i vari componenti software e quindi di creare una separazione tra la logica applicativa e l'interfaccia utente:
    - **Model:** fornisce le classi con i dati utili al programma e i vari metodi per accedervi;
    - **View:** rappresenta i "Form", visualizza i dati dei vari *Model* e quindi si occupa dell'interazione tra l'utente e la struttura sottostante;
    - **Controller:** riceve i comandi dall'utente tramite il *View* ed esegue operazioni che possono interessare il *Model* e solitamente cambia lo stato del *View*;

- **Singleton**: usato nella classe '*Database'* per garantire che l'istanza del database sia unica all'interno del programma;

- **Observer**: il suo soggetto (ISubject) è la classe '*Tavolo'*, ma l'Observer è usato nella classe '*PanoramicaTavoli*' e serve per "tenere sotto osservazione" e aggiornare, ogni minuto, il colore dei tavoli in base al tempo di attesa dei clienti nel ristorante. Con il metodo '*RefreshTavoli*' iscrivo tutta la lista dei tavoli all' Observer, poi con il metodo '*TimerTick'* ogni minuto viene fatto l'aggiornamento dei tavoli e mandata una notifica. Infine, quando un tavolo viene liberato o chiusa la finestra della panoramica, esso viene tolto dall'iscrizione dell'Observer.

4. Documentazione sull'utilizzo

- L'applicazione è stata realizzata su 'Windows 10 Home' usando il programma 'Visual Studio 2019'.
  Per il database è stato utilizzato Microsoft SQL Server Express, scaricabile al link: https://www.microsoft.com/it-it/sql-server/sql-server-downloads

  Mentre per l'interfaccia di gestione del database è stata usata l'applicazione 'Microsoft SQL Server Management Studio' versione 18.9.2. Scaricabile al link: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15

  Per eseguire l'applicazione si dovrà entrare nella cartella 'GestioneRistorante' e cliccare su 'GestioneRistorante.sln'. Successivamente si dovrà impostare tramite stringa di connessione l'indirizzo del database. Per fare questo, su 'Visual Studio' fare click destro sul progetto e andare su proprietà, poi su impostazioni e cambiare il valore dell'impostazione 'StringaConnessione' con la stringa ottenuta in fase di installazione del database.

- Prima di poter eseguire il software è necessario importare il database e tutte le tabelle. Per fare ciò è sufficiente eseguire lo script su 'Microsoft SQL Server Management Studio' contenuto all'interno della cartella 'ExportDB', chiamato 'SmartRestaurantDB.sql'.

5. Use Cases con relativo schema UML

L'utente, in questo caso il ristoratore, dopo aver aperto l'applicazione si ritroverà davanti la schermata della '*HomePage*' dalla quale potrà scegliere se creare una nuova prenotazione, visualizzare la lista delle prenotazioni già effettuate oppure la lista dei tavoli prenotati. Le prenotazioni posso essere di 3 tipi, asporto, domicilio e al tavolo, in quest'ultimo caso si potrà scegliere o meno di visualizzare il menù e ordinare cibi oppure confermare la prenotazione e ordinare nel momento in cui si arriverà al ristorante. Il ristoratore può inoltre visualizzare tutta la lista delle prenotazioni, modificarle e/o eliminarle. Quando il cliente dovrà pagare, cliccando sulla sua prenotazione verrà mostrato il riepilogo dell'ordine e il prezzo totale. Cliccando sul

pulsante "*Paga*" il tavolo viene liberato, ma la prenotazione non verrà eliminata, il suo stato verrà settato a 'Terminato' e sarà ancora presente nella lista delle prenotazioni, per un eventuale consultazione finito il servizio.