RELAZIONE

TAVERNPROJ

Giulia Lucchi
Eleonora Guidi

INDICE

Capitolo 1
Analisi

L’ idea di questo progetto è sorta dalla collaborazione fra le sottoscritte e l' Osteria Lucchi di Montiano. Il personale di quest'ultima ha sentito la necessità di possedere un modo più comodo per gestire le prenotazioni all' interno dell'attività e noi abbiamo colto l'occasione per creargli un programma che gli permettesse di farlo.

1.1 Requisiti

Il software ha come intento quello di fornire all'utente, o meglio al personale del ristorante, la possibilità di gestire le prenotazioni, in modo semplice e comodo. Il programma infatti ha come funzionalità principale l'inserimento, la modifica e la cancellazione di una prenotazione, con tutti i dati che ne conseguono. Inoltre l' introduzione della mappa del locale, dà l'opportunità di disegnare, controllare e coordinare la disposizione dei tavoli prenotati nel giorno corrente. Infine la presenza dei pulsanti sottostanti garantisce un modo facile e veloce per accedere alle prenotazioni.

In breve, le funzionalità che il software dovrà gestire sono :

- Inserimento di una nuova prenotazione, attraverso un calendario per la scelta del giorno
- Richiesta di un menu speciale o aggiunta di note al momento della prenotazione
- Modifica di una prenotazione scegliendo per giorno o per nome
- Cancellazione di una prenotazione scegliendo per giorno o per nome
- Controllo dello stato delle prenotazioni del giorno corrente, attraverso i bottoni adibiti. Oltre ad aprire la prenotazione corrispondente, questa può anche essere modificata o eliminata.
- Disegno della disposizione dei tavoli per la serata sulla mappa del locale.

1.2 Problema

Travernproj dovrà essere in grado di salvare le prenotazioni nel giorno scelto dall'utente e, se il giorno è quello corrente, dovrà porle in evidenzia, aggiungendo un apposito pulsante nella schermata iniziale. Se ci sono prenotazioni per il giorno corrente, l'utente sarà in gradi di inserire il tavolo sulla mappa, scegliendone la

posizione.
Un ulteriore problema è la persistenza dei dati all'interno del software: alla chiusura, si dovranno salvare tutte le prenotazioni effettuate, ma anche tutti i tavoli disegnati sulla mappa del locale; mentre, al momento dell' avvio, si dovranno ricaricare tutte le prenotazioni e disegnare i tavoli precedentemente conservati, se la disposizione salvata dei tavoli è quella del giorno corrente.
La realizzazione di un archivio contenente le prenotazioni, utile per redigere eventuali statistiche, sarà da effettuarsi tramite adeguato database, che verrà sviluppato in un secondo momento e non sarà per tanto presente in questa versione del progetto.

Capitolo 2
Design

2.1 Architettura

Il programma è stato sviluppato seguendo il pattern architetturale MVC (Model-View-Controller), per assicurare una separazione fra le parti del progetto e una miglior divisione dei compiti, controllo e manutenzione del software. (Figura 2.1)

2.1.1 Model

Il Model ha l'obiettivo di incapsulare lo stato dell'applicazione. Qui si definiscono le regole per l'interazione con i dati, implementando le funzionalità di accesso e aggiornamento delle prenotazioni e tutti i dati annessi ad esse. La struttura dati principale è un HashMap, utilizzata per il salvataggio delle prenotazioni. All' interno del Model troviamo anche la classe per la gestione del disegno, creato grazie alla libreria 'Graphics'. I tavoli possono essere creati nella cartina dal momento in cui l'utente ha inserito almeno una prenotazione in quel giorno. Essi sono identificati da un indice e delle coordinate.

2.1.2 View

La view rappresenta la parte più complessa e articolata del software.
Oltre alla schermata principale del software, agendo sui diversi pulsanti, siamo in grado di aprire nuove form per l'inserimento, la modifica e la cancellazione delle prenotazioni. Inoltre l'utente può avvalersi dell'uso di un calendario per la scelta delle date ogni volta vengano richieste.
Le classi Chooser ci consentono di decidere il metodo di modifica o cancellazione della prenotazione ,scegliendo per nome o per data.
E' possibile prendere visione in modo più dettagliato della view negli screenshot presenti in Appendice A.

2.1.3 Controller

La classe Controller ci da la possibilità di gestire tutte le interazioni che avvengono tra le classi della view e richieste di dati presenti nel model. Inoltre gestisce la persistenza dei dati, sia per quanto riguarda le prenotazioni che per il disegno della mappa, salvando su file alla chiusura dell'applicazione e

ricaricando da esso all'apertura.

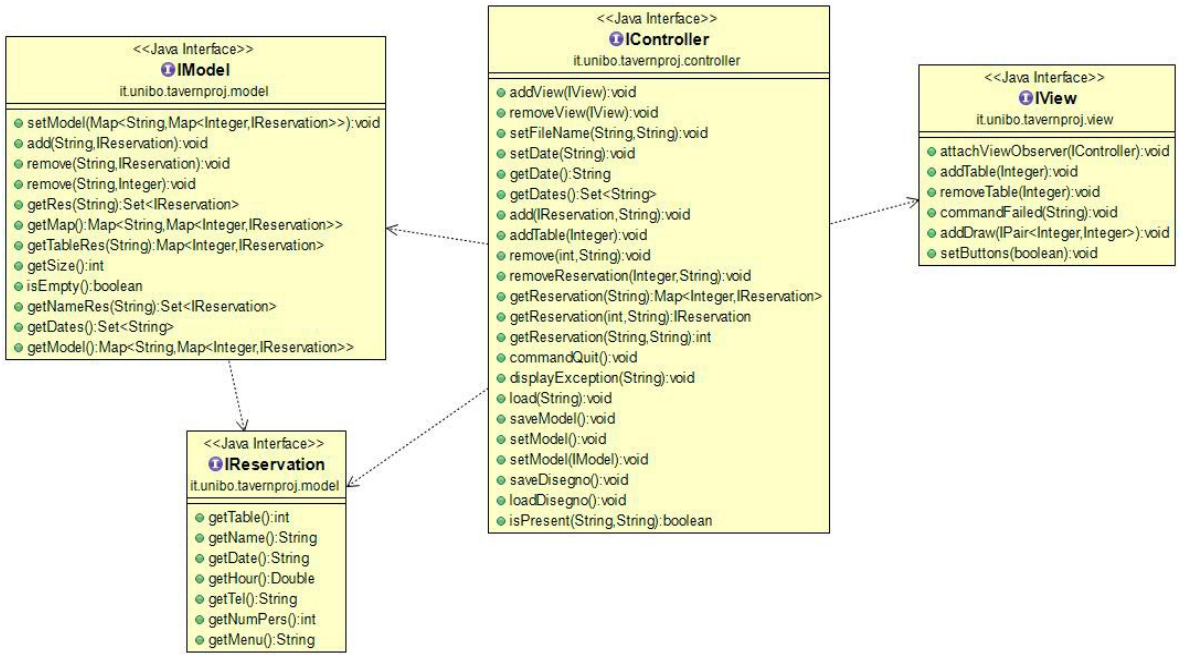


Figura 2.1: UML rappresentante lo schema MVC utilizzato nel software.

2.2 Design dettagliato

2.2.1 Organizzazione Package

- ***it.unibo.tavernproj.controller*** *:* contiene la classe Controller con relativa interfaccia IController. Gestisce tutte le interazioni tra view e model.
- ***it.unibo.travernproj.model*** **:** contiene le classi :

  -Reservation, che implementa l'interfaccia IReservation, ha come obiettivo di creare l'istanza della prenatozione. Le prenotazioni infatti vengono salvate come oggetti IReservation all'interno della mappa. Essa è composta da campi privati , che rappresentano i dati da salvare per ogni prenotazione. A questi campi si può accedere solo attraverso i getter, inseriti nella classe stessa.

  - Model, che implementa l'interfaccia IModel, contiene la struttura data in cui si salvano tutte le prenotazioni. La struttura dati in questione è un HasMap, che ha come chiave la data della prenotazione e come valore un' ulteriore

HashMap, avente, a sua volta, come chiave il numero del tavolo e come valore la 'Reservation'. Questa scelta è dovuta da una necessità, sorta a tempo di implementazione, per la gestione degli inserimenti e rimozione delle prenotazioni. Inoltre all'interno della classe sono presenti tutte i mari metodi per la gestione delle prenotazioni.(figura 2.2.1)

- ***it.unibo.tavernproj.model.disegno*** :

  - DrawMap contiene la struttura dati HashMap nella quale son salvati i tavoli disegnati nella cartina, identificati attraverso un indice crescente, chiave della mappa, abbinato alle coordinate dei tavoli, di tipo IPair. La classe non implementa alcuna interfaccia, poiché non l'abbiamo creata. Questa decisione è stata presa, in quando la classe utilizza il pattern singleton e quindi, oltre all'override dei metodi per la mappa, non ci sono altri metodi.

  -Pair, che implementa l'interfaccia IPair, è una classe generica che viene utilizzata per salvare le coordinate x e y dei tavoli disegnati nulla cartina, per compiere un azione di salvataggio.

  contiene la struttura dati in cui vengono salvati i tavoli disegnati e la classe generica in usata come oggetto in cui inserire le coordinate.

- ***it.unibo.tavernhproj.test*** : contiene alcuni test JUnit, che verificano la efficienza e la correttezza del Model.

- ***it.unibo.tavernproj.view:*** contiene le classi :
  - Application che contiene il main dell' applicazione e rappresenta la classe principale del sistema. (figura 2.2.2)
  - View e relativa interfaccia IView, rappresentanti le viste principali. (figura 2.2.3)
  -ButtonPanel utilizzata per costruire un pannello di bottoni sulla view principale e quindi richiamata da essa.

- ***it.unibo.tavernproj.view.calendar:*** contiene la classe Calendar e relativa interfaccia ICalendar, utilizzate per costruire e gestire il calendario dell' applicazione.

- ***it.unibo.tavernproj.view.disegno :*** contiene le classi :

  -DrawPosition, che implementa l'interfaccia IDrawPosition, in cui avviene la cancellazione e il disegno dei tavoli sulla cartina del locale. Contiene infatti un

due metodi per la cancellazione : uno per cancellare quello precedente e l'altro per cancellare tutto. Inoltre ci sono i metodi, che utilizzano la libreria Graphics, per disegnare concretamente, attraverso il drawRect(x,y,w,h). (figura 2.2.3)

- ***it.unibo.tavernproj.view.form*** : il package contiene le classi:
  - Acceptor, ProgressiveAcceptor, ProgressiveAcceptorImpl e ProgressiveFilter sono utilizzate per costruire ed aggregare i campi della form delle prenotazioni.
  - BasicFrame estende JFrame e costruisce un pannello con le dimensioni pari a metà di quelle dello schermo.
  - Chooser con interfaccia IChooser, che estendono BasicFrame e a loro volta estesi da ModifiedChooser, costruiscono una finestra per la scelta della modifica/cancellazione della prenotazione da parte dell' utente per nome o per data. (figura 2.2.4)
  - ReservationForm che implementa l' interfaccia IReservationForm ed estende BasicForm, estesa da NewReservationForm che implementa INewReservationForm, estesa a sua volta da ModifiedTableForm che implementa IModifiedTableForm, estesa a sua volta da DeleteTableForm, gestiscono tutte le form relative alle prenotazioni (inserimento, modifica e cancellazione di esse). (figura 2.2.5)
- ***it.unibo.tavernproj.view.utilities:*** in questo package sono contenute le classi di utilities. Sono classi contenente metodi usati perpendicolarmente in tutta l' applicazione. La classe principale è Utilities che implementa IUtilities. Tale classe è estesa da BasicGUIutilities, che implementa IBasicGUIutilities e contiene la maggior parte dei metodi per settare la grafica di base dei componenti principali del software (bottoni, pannelli…). Tale classe è a sua volta estesa da GUIutilities, che implementa IGUIutilities. (figura 2.2.6)

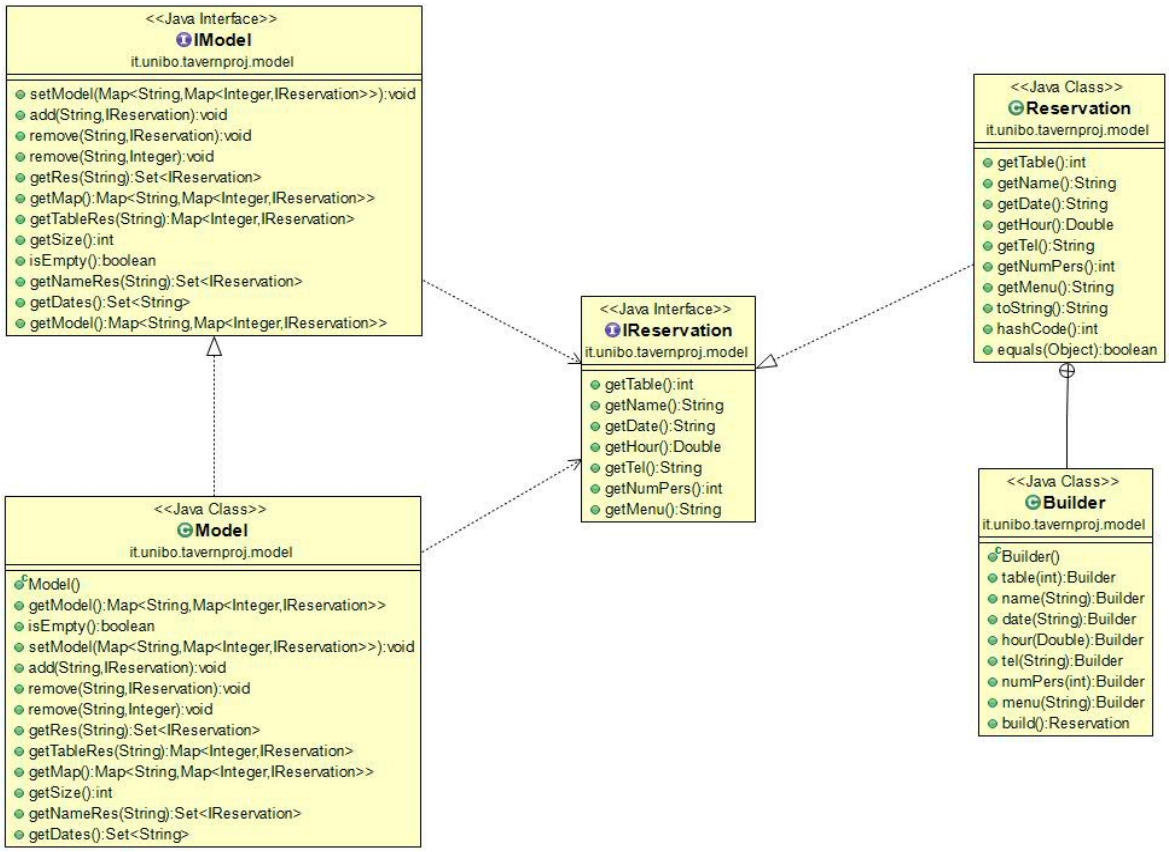


Figura 2.2.1: UML del modello

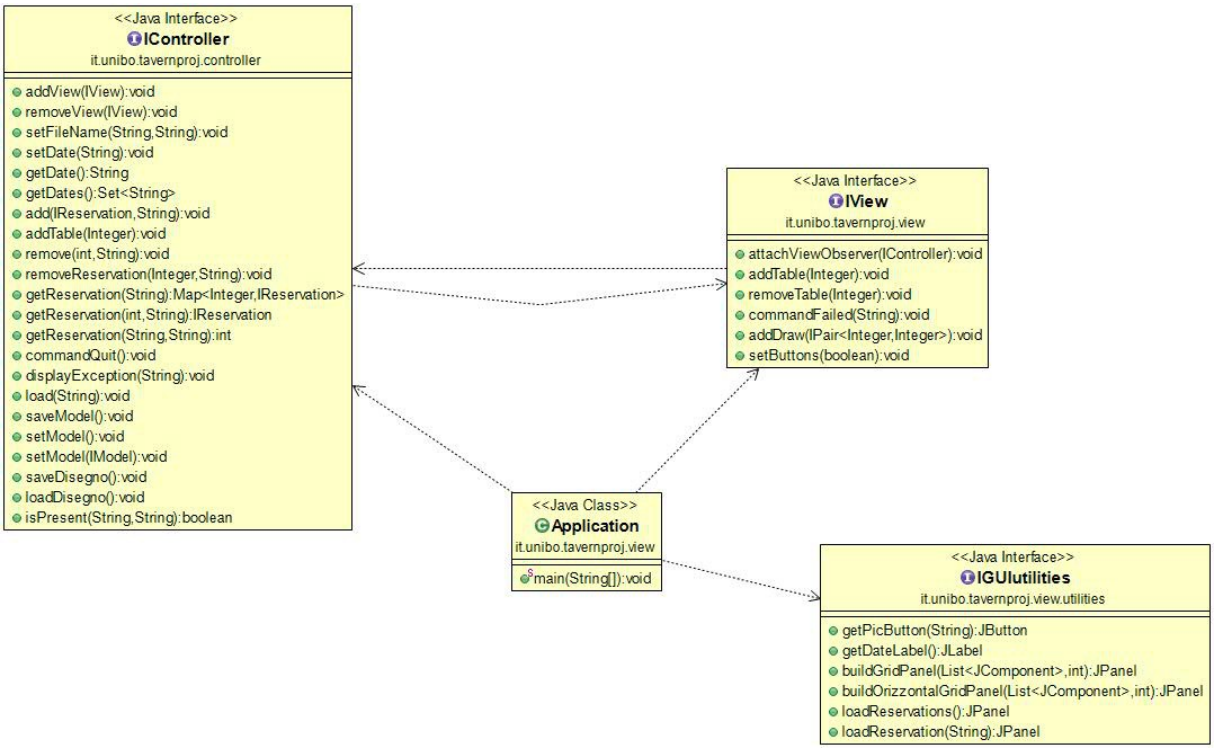


Figura 2.2.2 : UML dei componenti base dell' Applicazione

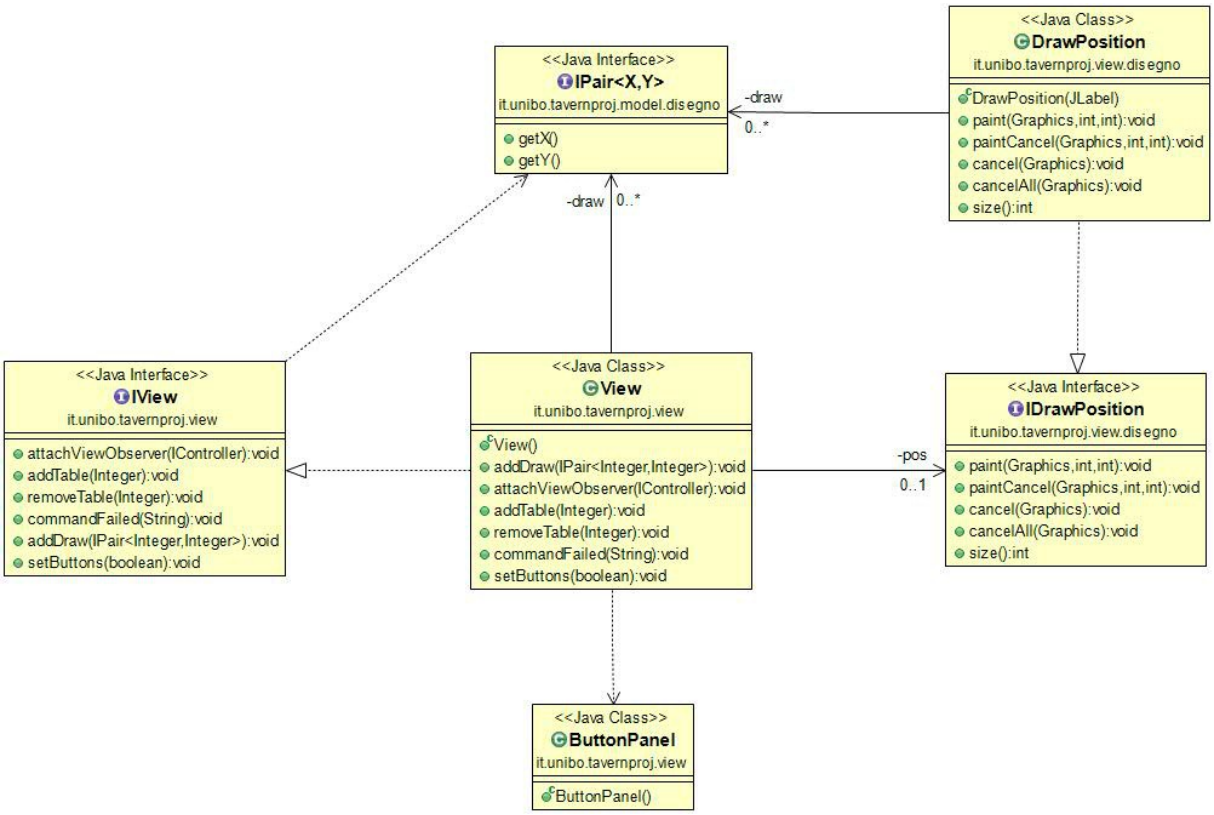


Figura 2.2.3 : UML della view con interconnessioni al sistema di disegno

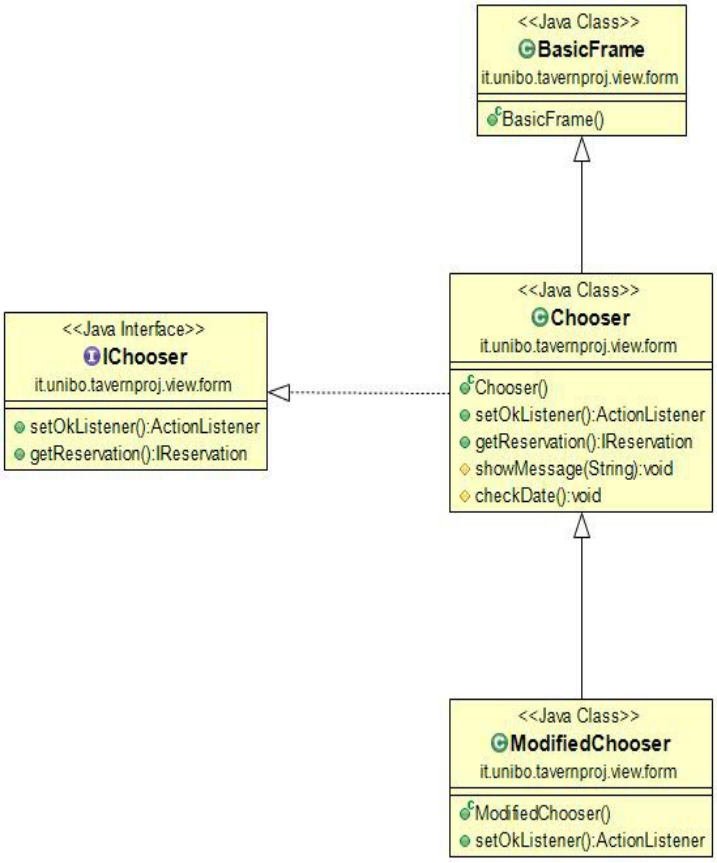


Figura 2.2.4: UML del Chooser e sua estensione

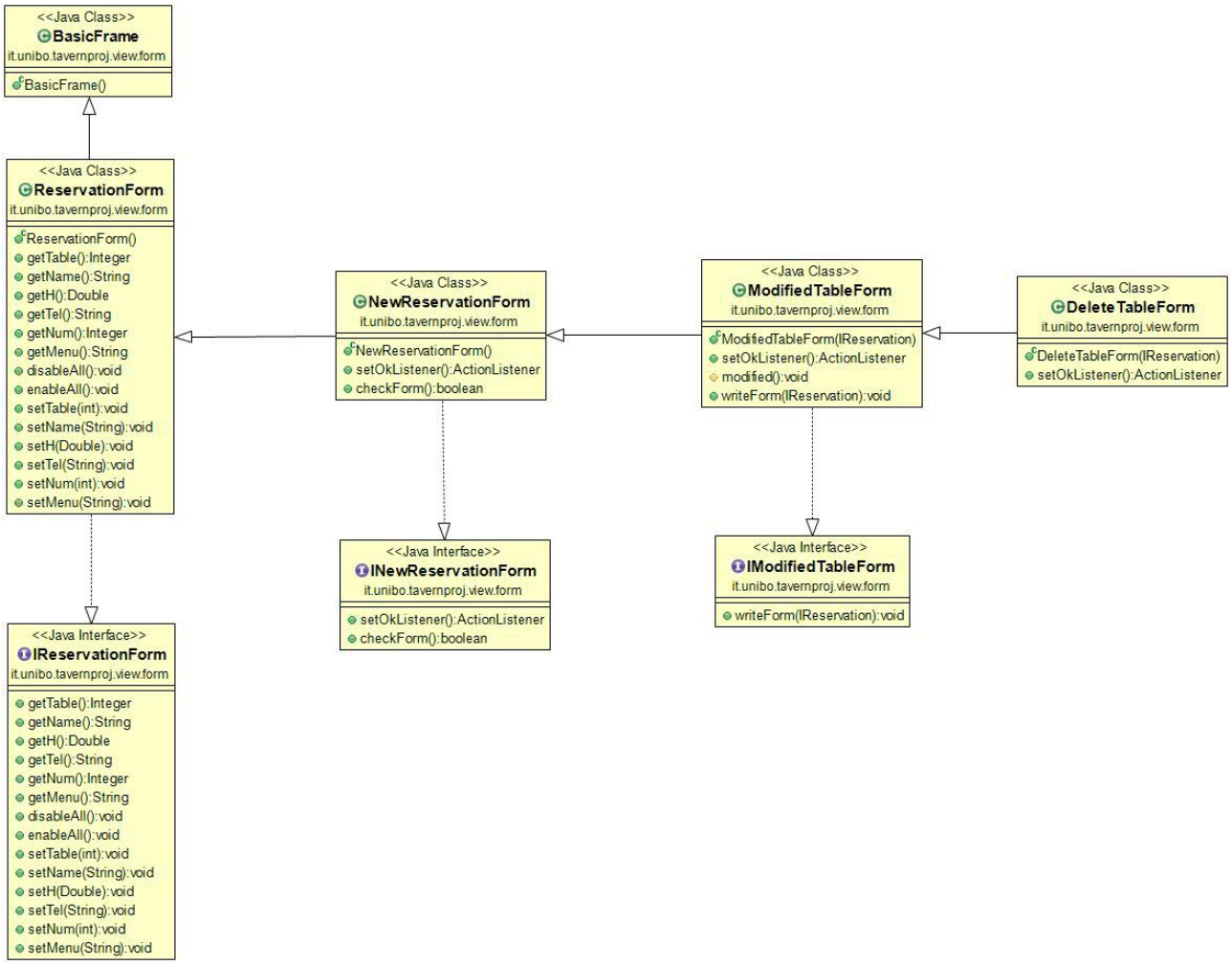


Figura 2.2.5: UML della struttura utilizzata per inserimento, modifica e cancellazione delle prenotazioni.

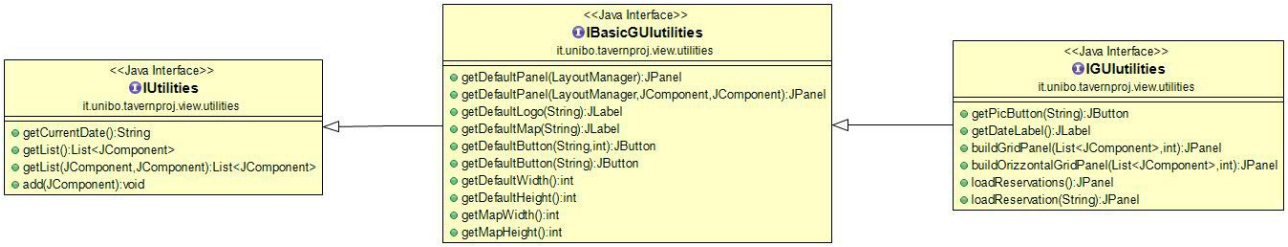


Figura 2.2.6: UML delle classi di utilities

2.2.2 Pattern Utilizzati

I pattern utilizzati nel progetto sono:

- Singleton: nella classe Controller e in quella DrawMap. Entrambi sono stati utilizzati per la tipica caratteristica del pattern che ci consente di istanziare un unico oggetto a fronte di chiamate multiple.
- Builder: nella classe Reservation. Abbiamo optato per questo pattern, implementato tramite una Inner Class Builder, in quanto il costruttore della classe Reservation prevedeva molti parametri.
- Decorator: nelle classi ReservationForm e Chooser. In entrambi i casi abbiamo optato per un decoratore che ci permettesse di fare l'override del metodo SetOkListener, in modo da gestire diversamente la sua funzione a seconda della classe, ma riutilizzando il più possibile il codice.

Capitolo 3
Sviluppo

3.1 Test Automatizzati

All' interno del progetto abbiamo utilizzato alcuni test automatizzati, che si occupano di verificare la corretta implementazione e funzionamento del model ed, in parte, anche del controller. Per questi test abbiamo utilizzato la suite Junit, per provare la creazione di una prenotazione con i relativi metodi. Inoltre abbiamo controllato anche la corretta implementazione della struttura dati utilizzata e di tutti i metodi riferiti a questa, che mirano alla gestione delle prenotazioni.
Ci è sembrato opportuno anche verificare i metodi di salvataggio e caricamento delle prenotazioni, per verificare l'efficienza e la correttezza di ciò.
Questi test sono stati molto utili per correggere e modificare ciò che era stato implementato.
Per quanto riguarda il testing della view e del controller, a esclusione del salvataggio e del caricamento, non abbiamo utilizzato dei test automatizzati, ma abbiamo testato manualmente e visivamente attraverso l'esecuzione del programma.
Ed infine abbiamo testato il software in diversi sistemi operativi, quali Windows Linux Mint e Mac.

3.2 Divisione dei compiti e metodologia di lavoro

Il lavoro è stato diviso fra gli autori del progetto seguendo principalmente il pattern architetturale MVC, come si può vedere nel paragrafo 2.1. :
Giulia Lucchi

- realizzazione del modello, che comprende la gestione dei tavoli e delle prenotazioni)
- realizzazione della gestione della cartina per il disegno dei tavoli
- salvataggio e caricamento dei dati per le prenotazioni
- salvataggio e caricamento dei dati per il disegno dei tavoli

Eleonora Guidi

- realizzazzione della view
- realizzazione della grafica, compresa la piantina del locale
- le funzioni utili a gestire le principali funzioni inerenti alla View

All'inizio abbiamo discusso insieme le interfacce da sviluppare, in modo da coordinarci meglio e organizzare le parti da implementare poi singolarmente. Insieme abbiamo composto e implementato il controller, per unire in modo armonico e coerente le due parti svolte singolarmente. Ovviamente questo ha portato a delle piccole modifiche, non invasive, di alcune parti.
In conclusione abbiamo fatto un check finale a progetto concluso.

Come DVCS abbiamo utilizzato Mercurial. Come richiesto, abbiamo creato un repository su Bitbucket al seguente indirizzo:

https://bitbucket.org/Eleonora_Guidi/oop14-tavernproj.

Entrambe abbiamo utilizzato costantemente Mercurial, sia da linea di comando che attraverso plugin per Eclipse.

3.1 Note di sviluppo

Essendo il primo progetto sul quale abbiamo lavorato, abbiamo seguito il consiglio del professore "si ricorda che agli ingegneri non `e richiesto di re-inventare la ruota continuamente" . Per questo per il nostro progetto ci siamo ispirati ad un programma fornitoci nella lezione del Laboratorio 8. Da lì abbiamo preso spunto per l'organizzazione e la progettazione del nostro progetto.

Abbiamo utilizzato anche due classi degli esami degli appelli scorsi, in quando ci ricordavamo, grazie al nostro esercizio a casa, di classi già fatte e abbiamo rimosso i metodi inutili al nostro progetto:

- la classe Pair<X,Y> del package esami2013.appello02.sol2, per il salvataggio delle coordinate dei disegni dei rettangoli per i tavoli.
- Le classi Aggregator<X>, ProgressiveAcceptor<X>, PRogressiveFilter<X> ProgressiveAcceptorImpl<X> dell'esame 01b del 2015. Queste sono state utilizzate per implementare la Form di base.

Il calendario all'interno del programma è stato preso da un programma già fatto e modificato per adattarlo alle nostre esigenze, aggiungendo un metodo con scopo di capire la correttezza della data scelta a fronte della modifica delle label dei giorni in Jbutton. Il link di riferimento è http://javabelazy.blogspot.it/2012/01/java-swing-

calendar.html.

Fra tutti quelli visti in rete era quello che si è sembrato più adatto. Esso è creato grazie alla libreria, già conosciuta, Java Swing.

Per l' inserimento delle immagini nei bottoni mi sono basata su questo: http://stackoverflow.com/questions/299495/how-to-add-an-image-to-a-jpanel

Capitolo 4
Commenti Finali

4.1 Conclusioni e lavori futuri

Il progetto è stato svolto in modo, secondo noi, accurato e organizzato seguendo una buona logica. Abbiamo cercato di applicare tutti i suggerimenti e regole sentite a lezione. A nostro parere, abbiamo prodotto un software abbastanza chiaro e intuitivo, dal punto di vista dell'utente. Un altro punto di forza può trovarsi nel fatto che il programma risulta facilmente applicabile in qualsiasi contesto di ristorazione, cambiando facilmente il logo e la mappa all'interno del programma.
Inoltre, come dichiarato ad inizio relazione, il programma è facilmente utilizzabile come interfaccia grafica per l'interazione con un database da realizzare. In tal modo saremmo in grado di consultare efficientemente le prenotazioni e trarre eventuali statistiche.

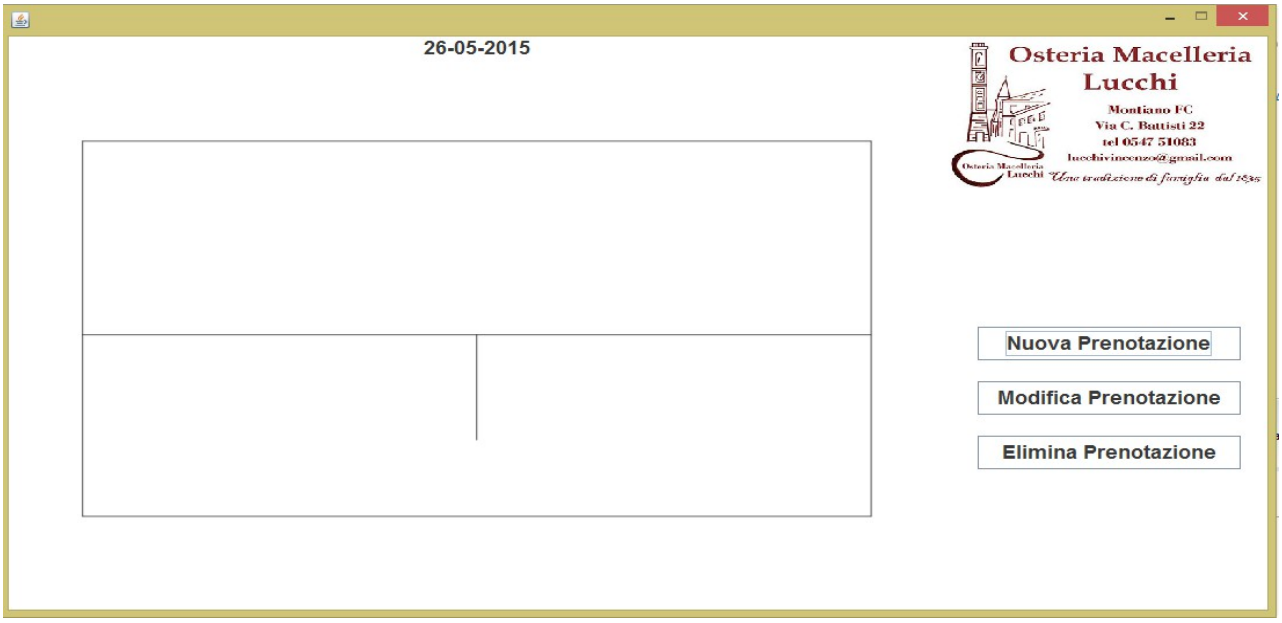
4.2 Difficoltà incontrate e commenti per i docenti

Nel corso del progetto siamo incorse nel bug 446691 di Eclipse che ci ha portato via parecchio tempo in quanto bloccava completamente il programma. Dopo decine di tentativi l' unico metodo di risoluzione trovato è stato quello di rimuovere tutte le lambda presenti nella classe Calendar. Per questo nella classe non sono presenti lambda come invece nel resto del codice.

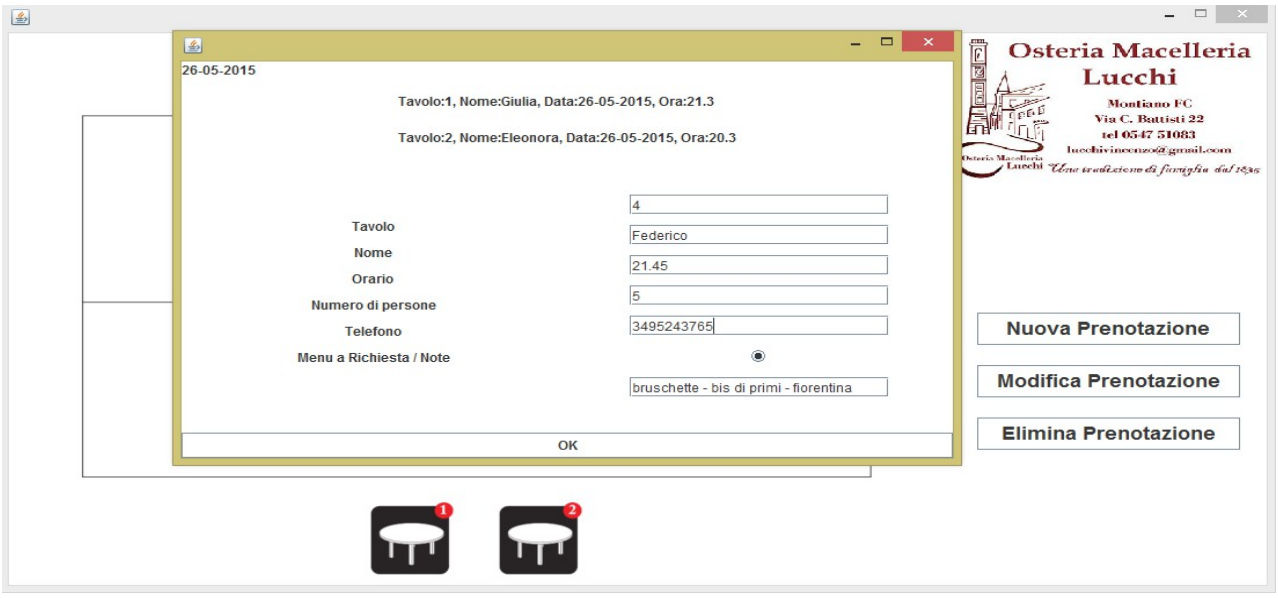
APPENDICE A
GUIDA UTENTE

Avviando il programma si apre una schermata avente la data corrente, il logo del locale e una cartina dall'alto del locale. A fianco della cartina ci sono le tre funzioni principali:

- nuova prenotazione : aggiunge la prenotazione
- modifica prenotazione : modifica la prenotazione già esistente
- elimina prenotazione : elimina prenotazioni esistenti

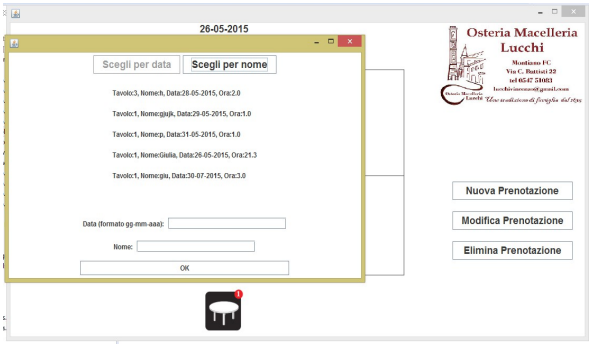


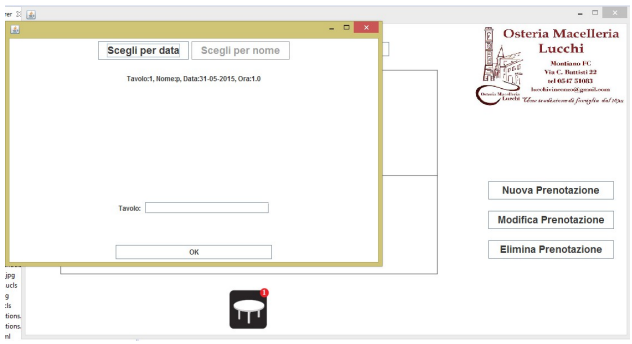
NUOVA PRENOTAZIONE



Premuto il pulsante 'Nuova Prenotazione', si apre un calendario nel quale si sceglierà il giorno in cui salvare la prenotazione. Scelta la data giusta, si aprirà una finestra avente tutti i dati da inserire nella prenotazione e si dovrà scrivere negli appositi spazi le informazioni della prenotazione.
Se la prenotazione avesse richiesto un menù fisso o qualsiasi altra necessità allora c'è una spunta da selezionare per far comparire la casella di testo in cui annotare ciò che serve (come si nota in figura).

MODIFICA PRENOTAZIONE e ELIMINA PRENOTAZIONE

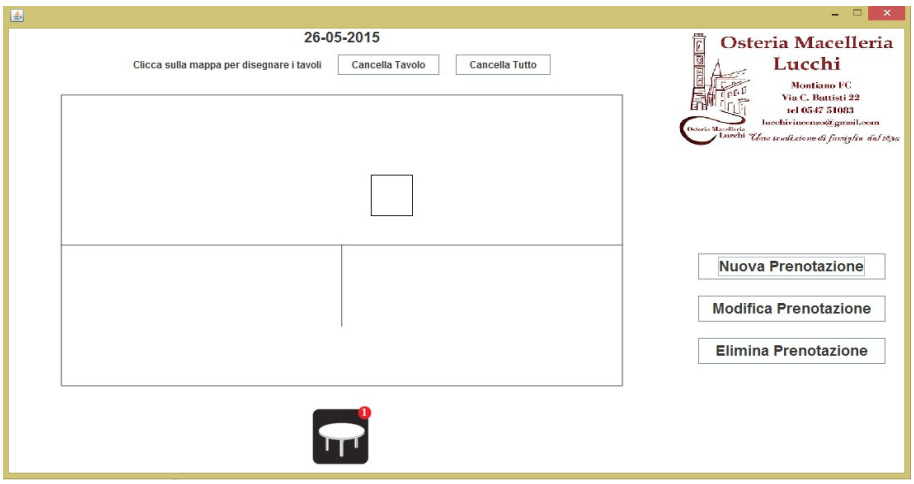




Premendo il pulsante 'Modifica Prenotazione', l'utente si ritroverà davanti a due possibilità di ricerca della prenotazione da cambiare/eliminare : ricerca per nome o per data. Con la ricerca per nome, il programma per modificare/eliminare richiede la data e il nome della prenotazione da modificare/cancellare, mentre con la ricerca per data basta inserire il numero del tavolo.

DISEGNO DEI TAVOLI

Dal momento in cui ci sono delle prenotazioni per il giorno corrente, escono nella schermata i pulsanti per la cancellazione dei tavoli che si possono disegnare nella mappa. Il disegno dei tavoli avviene tramite un semplice click. Con il pulsante 'Cancella Tavolo' viene cancellato l'ultimo tavolo disegnato e con 'Cancella Tutto' tutti i tavoli disegnati sulla cartina.



PULSANTI DELLE PRENOTAZIONI DEL GIORNO

Le prenotazioni del giorno corrente vengono visualizzate sulla schermata principale tramite delle icone di tavoli. Queste icone sono pulsante, per cui se l'utente clicca sul bottone si aprirà una finestra con le informazioni della prenotazione e la possibilità di rimuove o modificare i dati.