Corso di Programmazione ad Oggetti

A.A 2015-2016

Relazione progetto **Beach-Manager**

Autori:

Gianluca Iaia

Alessandro Cornacchia

Indice

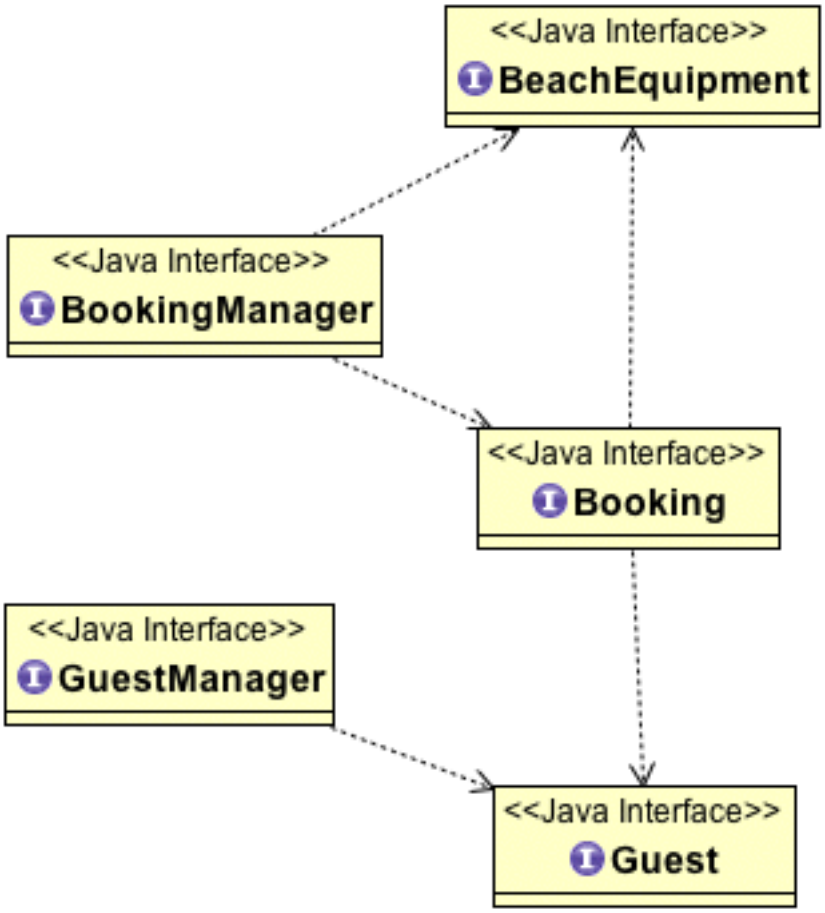
Capitolo 1
Analisi

1. Requisiti

Lo scopo del progetto è quello di realizzare un software di tipo gestionale per stabilimenti balneari, in modo da semplificarne la gestione e l'organizzazione. L'applicazione dovrà essere in grado di:

- Aggiungere e rimuovere prenotazioni;
- Permettere all'utente di visualizzare in modo intuitivo lo stato attuale (o futuro) della spiaggia;
- Registrare clienti;
- Assegnare ad ogni prenotazione un determinato cliente oppure, quando non ha importanza un cliente generico;
- Ottenere prezzi dei relativi servizi modificati in base al periodo stagionale e alla posizione a seconda della vicinanza dalla riva;
- Permettere all'utente di modificare facilmente la visione della propria spiaggia(modificare numero file e numero passerelle);
- Visualizzare dei grafici per mostrare un riassunto chiaro riguardante i guadagni.
- I dati devono essere persistenti, ciò significa che ad ogni apertura del programma i dati devono corrispondere all'ultima configurazione.

2. Analisi e modello del dominio



Il dominio applicativo del software è la gestione di uno stabilimento balneare. Dovendo modellare sia la gestione delle prenotazioni degli ombrelloni sia la gestione di un archivio dei clienti registrati, si è optato per avere due entità adempienti ai suddetti compiti, rispettivamente BookingManager e GuestManager. BookingManager è in grado di associare a ciascuna postazione spiaggia le relative prenotazioni. Per ‘postazione spiaggia' si intende una zona fissa di spiaggia assegnata ad un determinato cliente in un determinato periodo. Si comprendono quindi ombrelloni, gazebo, capanne o servizi di questo genere, si escludono invece lettini, sedie e sdraio. BookingManager agisce quindi su una o più Booking, e principalmente deve saper dire quando, da chi, o cosa, una postazione è

occupata, oltre che memorizzare, rimuovere e modificare le Booking presenti in un certo momento ed essere interrogato circa lo stato dell'archivio.

Booking contiene tutte le informazioni relative alla prenotazione, quindi il servizio richiesto, l'intestatario, le date di inizio e di fine del soggiorno.

Devono quindi essere modellate in primis le varie attrezzature/servizi che la spiaggia mette a disposizione, BeachEquipment è la corrispondente entità. Lettino, Ombrellone, Sdraio, ecc quelle che la specializzano.

GuestManager è l'analogo gestore dei clienti. Guest è un cliente della spiaggia, rappresenta un subset di una generica Persona, quindi oltre che ai dati anagrafici, possiede altre peculiarità utili al dominio applicativo in questione.

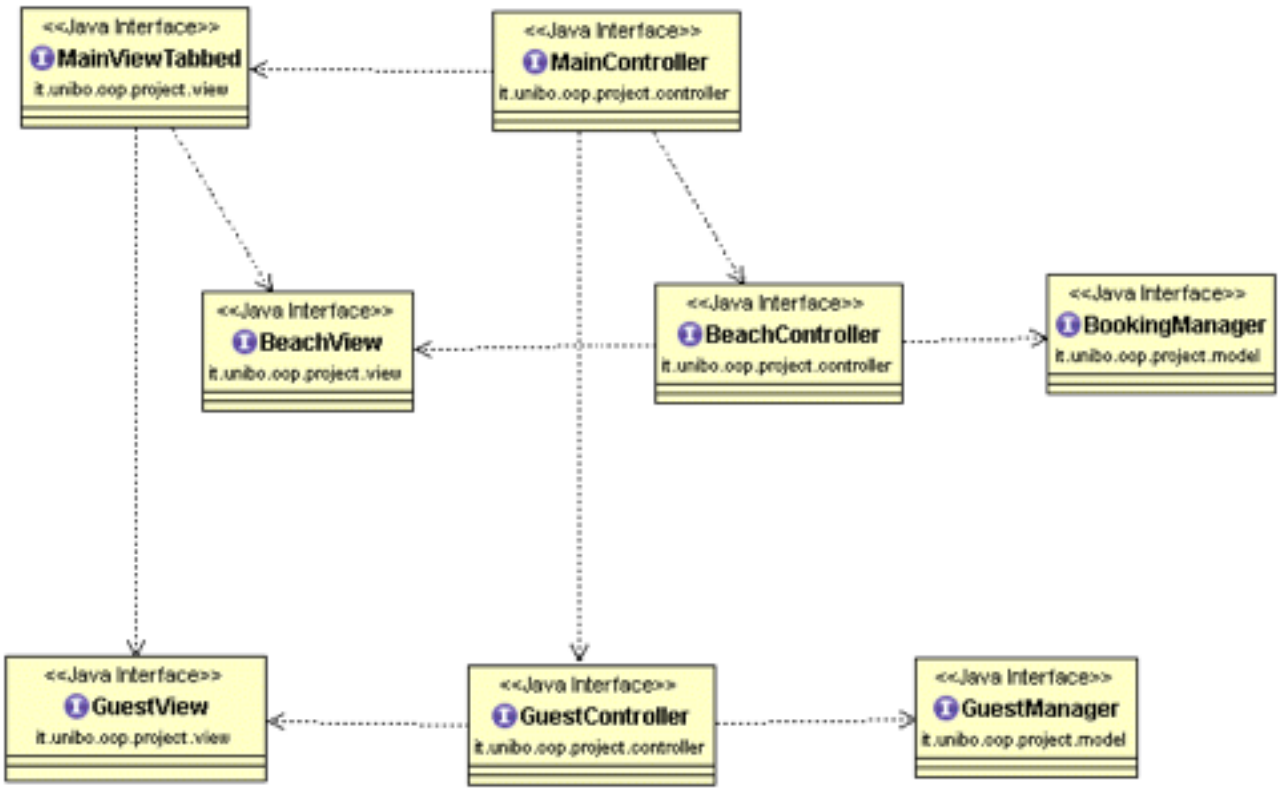
Si era pensato di introdurre anche una modellazione per servizi spiaggia che non fossero meramente ombrelloni e lettini, ad esempio noleggio imbarcazioni, menu pranzo, massaggi, ecc.. Si poteva quindi gestire in qualche modo una sorta di conto per ogni cliente registrato, che tenesse traccia appunto dei consumi su tali servizi, tuttavia per non superare il monte ore si è ritenuto che non fosse un requisito indispensabile, anche se la struttura complessivamente potrebbe permettere una eventuale integrazione in futuro.

Capitolo 2
Design

2.1 Architettura

Il progetto è stato sviluppato seguendo il pattern architetturale MVC, in maniera tale da rendere indipendenti il più possibile la parte grafica ed il modello che costituisce il core dell'elaborazione. Il Controller si occupa di coordinare le due parti sopracitate, interagendo con l'utente tramite la View,

nel nostro caso GUI, e gestendo le sue scelte tramite chiamate successive su metodi del Model. Abbiamo seguito questo pattern poiché, oltre a semplificare la divisione dei compiti tra i vari componenti del gruppo, offre un'elevata estensibilità al sistema.



In particolare ogni GUI è coordinata dal rispettivo controller, il quale viene chiamato dalla view associata, quando dall'esterno arrivano input o è necessario aggiornare dati. La view notifica i cambiamenti al controller che di conseguenza interroga il modello, il quale produce risultati, successivamente il controller setta i dati prodotti nella rispettiva interfaccia grafica.

Il controller accede al modello sostanzialmente in due entry-point: BeachController accede tramite BeachManager e analogamente il GuestController sfrutta il GuestManager. Tuttavia BeachController per avere una rappresentazione ad oggetti della struttura dello stabilimento usa anche l'entry-point Beach, si veda design di dettaglio del modello.

Abbiamo cercato di rispettare le politiche dell' MVC dove, la view non esegue calcoli, ma delega tutto al controller. Il model non conosce l'esistenza dell'interfaccia grafica, il controller si avvale semplicemente dei metodi pubblici del model per estrapolare(o settare) dati del modello in

modo che, se cambiando o modificando l'interfaccia grafica le modifiche da fare saranno solo nel controller, e saranno minime.

2.2 Design dettagliato

Model.

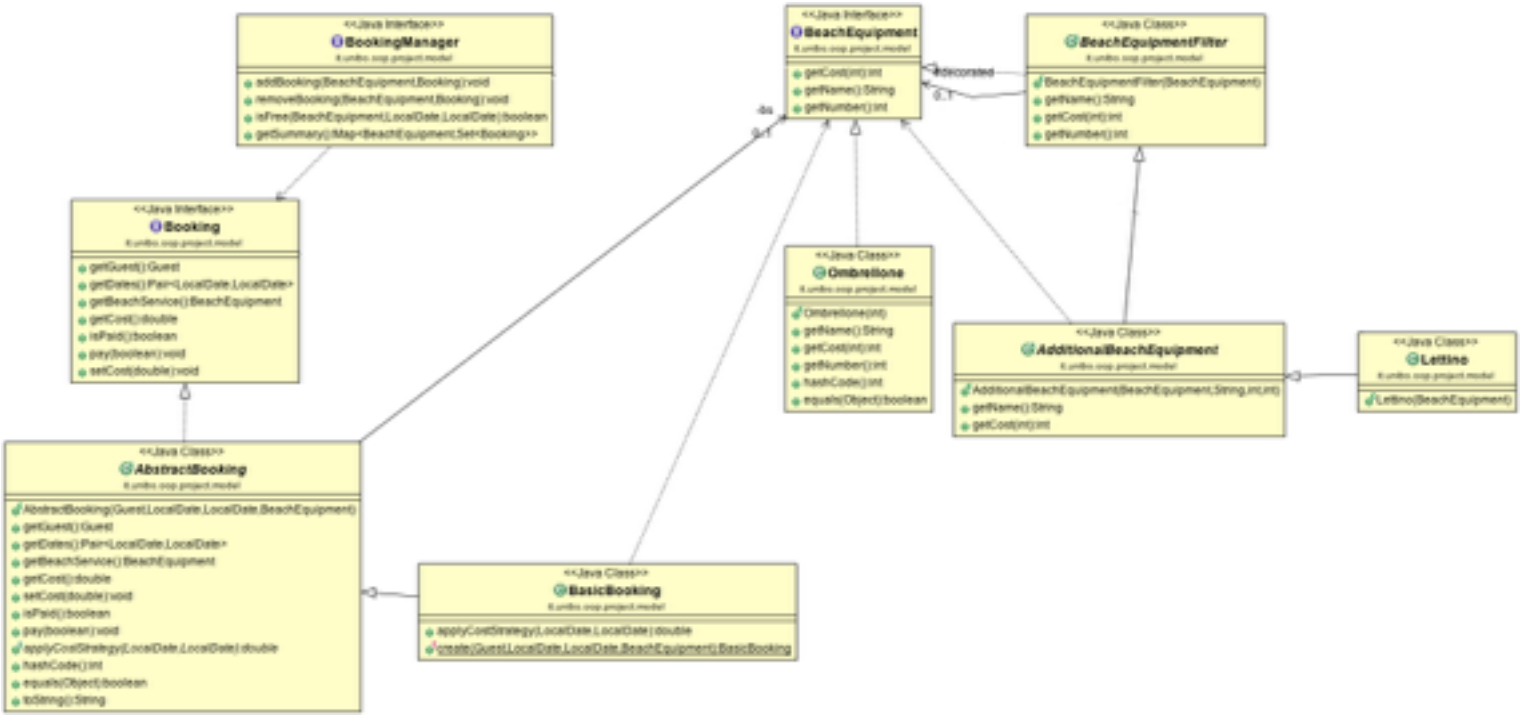


Fig 2.2.1

Fig 2.2.2

I servizi della spiaggia fruibili sono componibili tra loro, ma entro certi limiti. Per esempio un ombrellone non può essere combinato ad un gazebo, perché altrimenti rappresenterebbe due prenotazioni e non una. Viceversa, un ombrellone può essere abbinato ad uno o più lettini. Pertanto, si è deciso di sfruttare il pattern Decorator che intrinsecamente si presta molto bene alla soluzione di un problema di questa natura. Era anche possibile usare una enumerazione per rappresentare i vari servizi della spiaggia, ma in quel caso sarebbe stato più arduo controllare che venissero rispettati i vincoli di combinazione dei servizi.

Di conseguenza, dal momento che ci sono classi che modellano i vari BeachEquipment, si può facilmente modellare anche la struttura della spiaggia in una classe a sé stante, affidando, per il Single Responsibility Principle, a ciascuna classe un solo compito. Così si è preferito avere un'entità Beach piuttosto che accorpare questa logica al BookingManager.

Attualmente l'implementazione è molto semplice e minimale, ma potrebbe essere complicata a piacere in modo agevole senza andare ad impattare drasticamente altre classi del modello, e comportando solo poche modifiche a livello di controller.

L'interfaccia Booking è implementata dalla classe AbstractBooking che di fatto serve ad ospitare un Template Method, utile per il calcolo del sovrapprezzo in base al periodo della stagione. Nello specifico c'è solo BasicBooking che estende da questa classe astratta, ma aver impostato cosi la gerarchia facilita eventuali cambiamenti o aggiunte future. Nell'interfaccia Booking è presente il pattern creazionale Static Factory , più espressivo rispetto al semplice costruttore e di fatto utile agli altri sviluppatori del team per procedere senza sapere quale oggetto di quale classe viene creato.

Un'altra gerarchia Interfaccia + Classe astratta + Classe concreta è stata messa in campo per quanto riguarda la modellazione dei clienti. Siccome non è pensabile che lo stabilimento balneare registri ogni singolo cliente, compresi quelli che soggiornano un solo giorno, già a livello di modello si è distinto il cliente registrato che implementa anche l'interfaccia Persona, (GuestImpl) dal cliente generico (TemporaryGuest). I comportamenti comuni sono stati fattorizzati in AbstractGuest. Siccome la registrazione dei dati anagrafici richiede vari controlli e molti campi, si è usato il pattern Builder.

Quasi tutte le classi del modello offrono setter per permettere al controller di ripristinare lo stato del sistema dopo essere andato su file, quindi il modello né va su file, né chiama esplicitamente il controller, del quale non conosce neanche l'esistenza, perciò ne è del tutto indipendente.

Personalmente sono abbastanza soddisfatto del lavoro svolto.

View.

Ogni interfaccia grafica implementa l'interfaccia "View", la quale definsce di default alcune operazioni basiche che ogni view deve mettere a disposizione, ad esempio mostrare messaggi di errore o di varia natura. Oltre ad estendere da View ogni interfaccia grafica viene gestita da "MainViewTabbed" che ha il compito di rimpiazzare o cambiare il tab corrente.

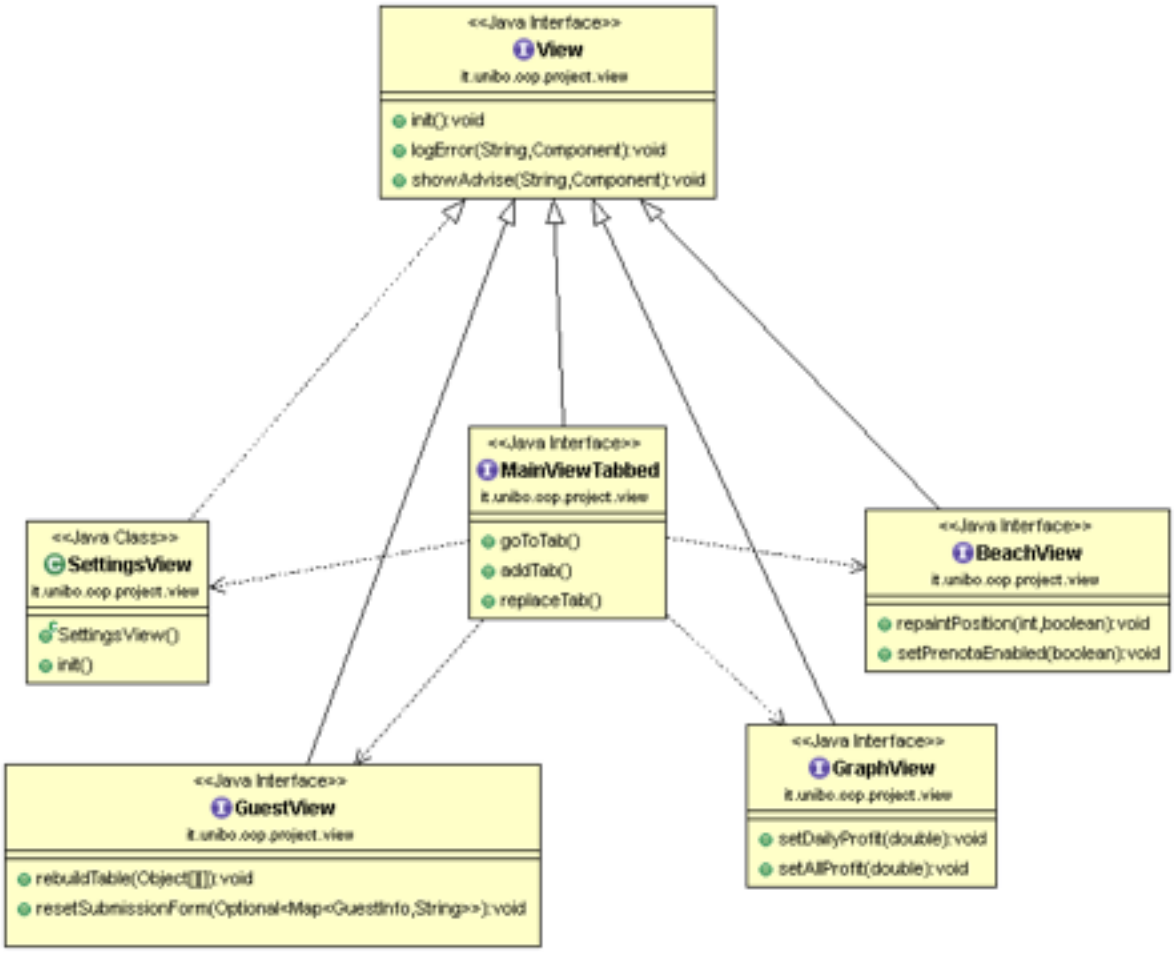


Fig 2.2.3

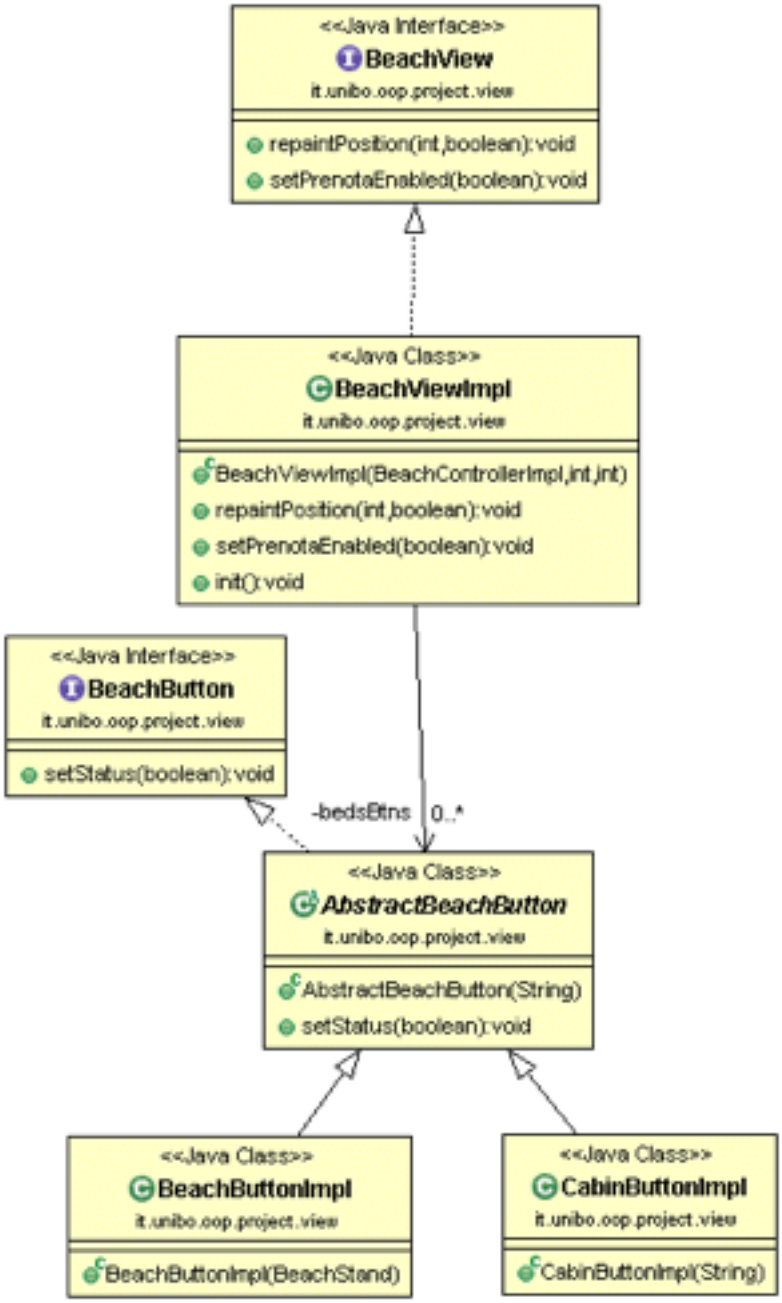
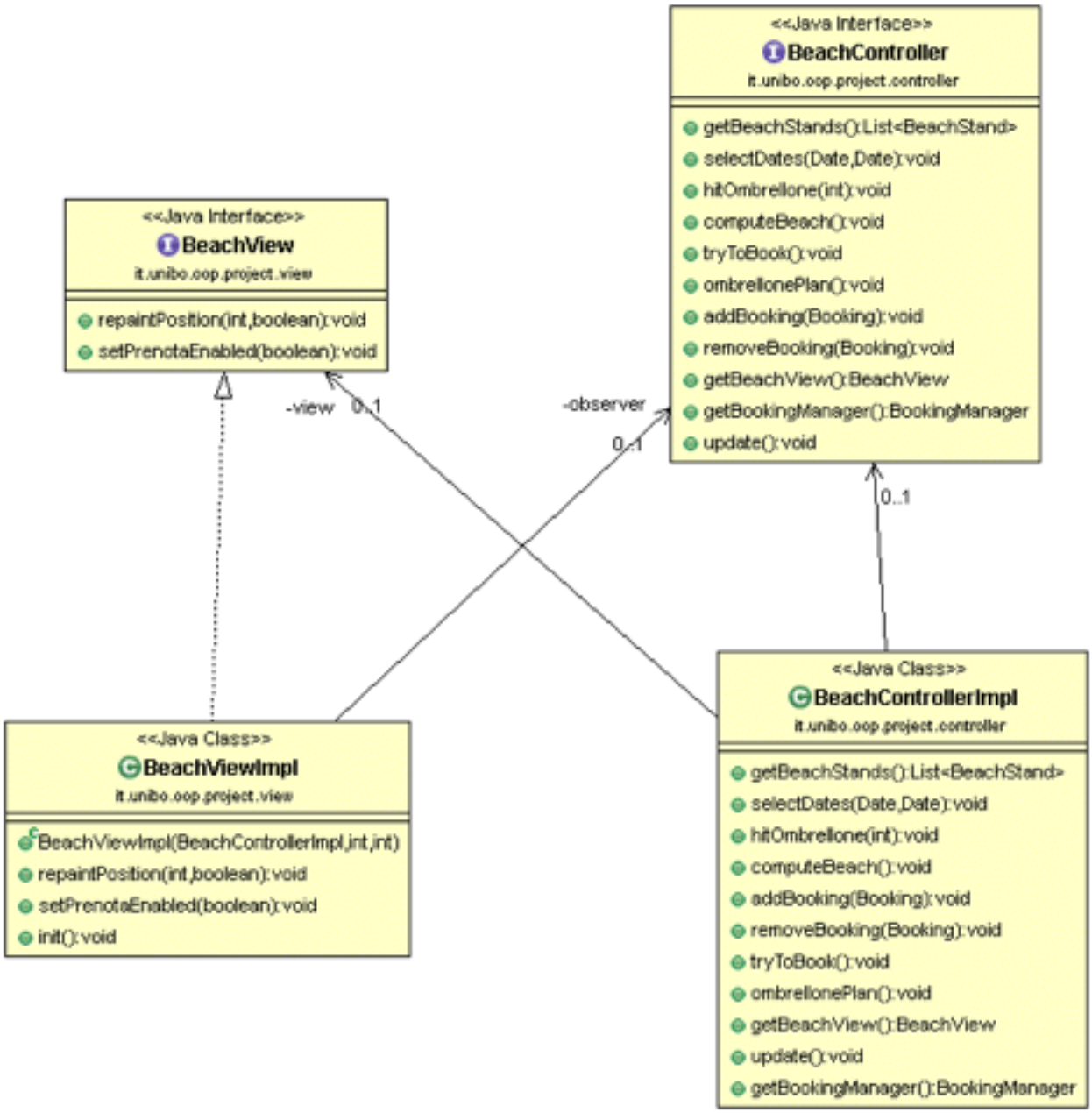


Fig 2.2.4

Ad ogni interfaccia è collegata la relativa implementazione.

L'interfaccia BeachView, viene implementata da BeachViewImpl che si occupa di gestire tutta la parte relativa alla visione della spiaggia, il ricalcolo dei bottoni rappresentanti le postazioni. Per rendere il più generico possibile e per evitare riscritture di codice, viene fattorizzato il concetto di "BeachButton" poiché ogni bottone ha in comune uno stato (libero o

occupato, rappresentato dai due colori verde e rosso), ed un testo, quindi l'unica cosa che differisce è l'immagine che dovrà rappresentare il relativo servizio.



Ogni View registra come "osservatore" il rispettivo controller, questa scelta è stata presa poiché far gestire tutto ad un unico controller poteva portare ad una "God-class" e quindi anche di difficile comprensione. La view offre al controller dei metodi pubblici per cambiare il proprio stato, così che quando notifica un evento, il controller dopo aver elaborato la risposta in base al Model, cambierà lo stato della View. Per limitare le possibilità di scelta dell'utente ed evitare input sbagliati in momenti sbagliati, le varie view al ricevere certi eventi disabilitano alcuni componenti.

Controller.

Il problema principale è stato il salvataggio su file della configurazione delle impostazioni, in modo da rendere il sistema persistente. Abbiamo discusso a lungo sulle varie possibilità, infine abbiamo deciso di delegare il salvataggio/caricamento da file al controller, poiché non ci è sembrata un’entità propria del modello. Questa scelta secondo noi è stata positiva poiché in questo modo il controller si occupa di prelevare dati da file in fase di inizializzazione, dopo averli caricati correttamente allora passerà i dati acquisiti al modello che li dovrà elaborare.

Capitolo 3
Sviluppo

3.1 Divisione dei compiti e metodologia di lavoro

Il progetto è stato inizialmente proposto come composto da un team di tre persone. La suddivisione dei compiti era stata definita in questo modo:

-Model: Alessandro cornacchia

-View: Gianluca Iaia

-Controller: Christian Bonanni

Successivamente però Bonanni non ha potuto procedere come stabilito, quindi tramite suggerimento del Prof.Viroli, abbiamo deciso di ripartirci il lavoro e di completare il progetto in questo modo:

-Model + BeachController, SettingsController e salvataggio file: Alessandro Cornacchia

-View + GuestController, GraphController: Gianluca Iaia

Inizialmente, dopo una prima fase di macro-progettazione del sistema, nella quale era stata decisa a grandi linee la struttura e scritto il codice per le interfacce principali, ogni elemento del gruppo lavorava in parallelo tramite l'ausilio di un DVCS (Mercurial). Il progetto procedeva con una buona indipendenza tra i tre sviluppatori, a volte facendo qualche passo indietro e rivalutando le decisioni prese nelle fasi precedenti, secondo un modello di sviluppo "agile" per tentare di arrivare velocemente ad una versione funzionante. In particolare alcuni metodi di Controller e View sono stati modificati nel momento in cui il gruppo si accorgeva che per gestire bene il sistema era necessario avere più controller di quanti inizialmente erano stati

preventivati. Poi però quando Bonanni non ha potuto più procedere , analizzato il lavoro nullo da lui svolto, abbiamo dovuto ripartirci quanto gli spettava e collaborato nella realizzazione del controller per raffinamenti successivi. Abbiamo infine ultimato il sistema in laboratorio cercando di sistemare i bug rimasti e rifattorizzando quegli aspetti che ci sembravano migliorabili.

3.2 Note di sviluppo

Le librerie utilizzate sono:

- JFreeChart e JCommon: per i grafici.
- JCalendar: per il calendario(componente grafico)
- SnakeYaml : per il salvataggio su file in formato yml, di facile gestione sia manuale che con script.
- Commons-IO: per facilitare l'I/O in fase di "installazione" .

Inoltre la classe ImagesConfiguration nel package View.configuration è frutto di uno spunto preso da un articolo simile su StackOverflow.

Capitolo 4
Commenti finali

4.1 Autovalutazione e lavori futuri

Terminato l'elaborato abbiamo notato che vi sono margini di miglioramento in esso, come una gestione più raffinata della struttura della spiaggia, incapsulata dentro la classe BeachImpl e al momento molto "naive", oppure dare la possibilità all'utente di impostare il tariffario, con i prezzi di ogni servizio per determinati periodi. Attualmente invece vi sono solo costanti dentro i vari BeachEquipment, che portano ad avere prezzi più o meno realistici.

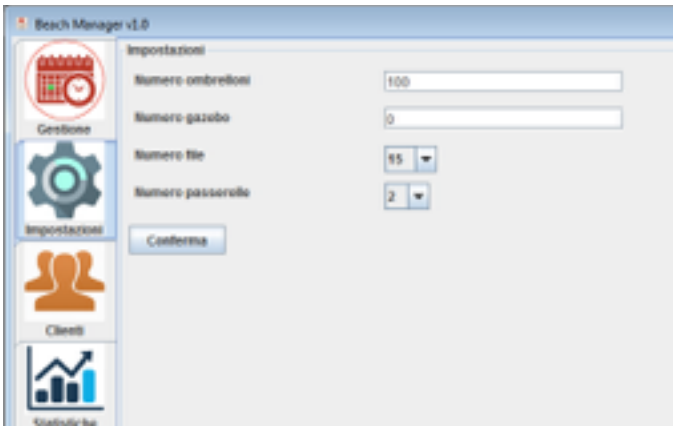
In definitiva, ci riteniamo piuttosto soddisfatti circa il lavoro svolto, ci è sembrato di aver implementato Model Controller e View indipendenti e autoconsistenti, cercando di usare al meglio gli strumenti che avevamo a disposizione e quanto conoscevamo. Forse molte cose in questo tipo di software sarebbe stato più facile realizzarle tramite una base di dati, ma non essendo ancora esperti in materia non ci siamo addentrati.

4.2 Difficoltà incontrate e commenti per i docenti

Da questa esperienza ci siamo accorti che il volume del progetto è stato troppo ampio per due persone, probabilmente in tre sarebbe stato ottimale. Abbiamo constatato inoltre tutte le difficoltà del lavoro in team, di conseguenza abbiamo apprezzato come non mai l'utilità delle interfacce come contratti d'uso delle entità in gioco, anche se è da ammettere che essendo il nostro primo progetto non è stato facile progettare e prevedere fin da subito quale fosse il modo migliore di organizzare il tutto, considerato anche che questo è tanto più vero quanto aumentano i requisiti del committente. Abbiamo inoltre riscontrato alcune difficoltà con l'utilizzo del DVCS.

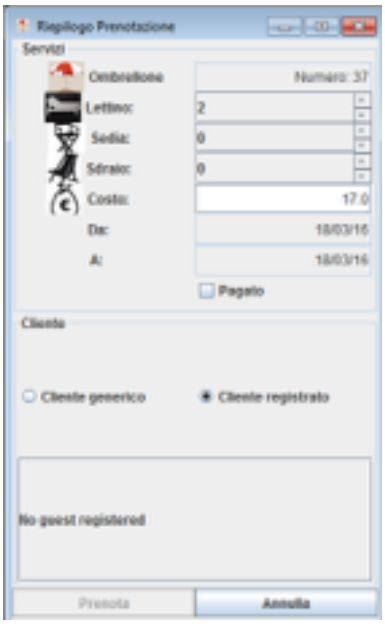
Appendice A
Guida utente

Al primo avvio del programma verrà visualizzata una visione di una spiaggia di default, per cambiare ed adattare la spiaggia visualizzata a quella reale, bisognerà recarsi nel tab a sinistra “Impostazioni”.



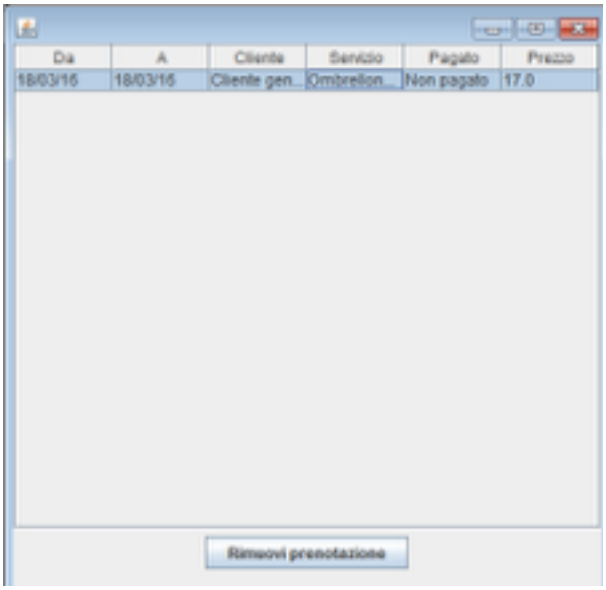
Tramite le caselle di testo si potranno modificare il numero di ombrelloni di cui la spiaggia dev’essere composta, il numero di gazebo, il numero di file (in quante file sono suddivisi gli ombrelloni) e il numero di passerelle. Completate le impostazione al click su “Conferma” verrà visualizzata la nuova visione.

Giunti nella visione “Gestione” si vedranno degli ombrelloni numerati verdi e rossi, quelli verdi indicano la disponibilità per il giorno corrente, per prenotare un ombrellone per giornata odierna basterà cliccare due volte su un bottone verde o cliccare a destra sul bottone “Prenota selezionato”. Se si vuole aggiungere una prenotazione a lungo termine bisognerà selezionare “Da-A” in alto a destra e scegliere le date desiderate, successivamente cliccare visualizza in modo da vedere la situazione della spiaggia in quel determinato giorno, poi successivamente cliccare sull’ombrellone desiderato.



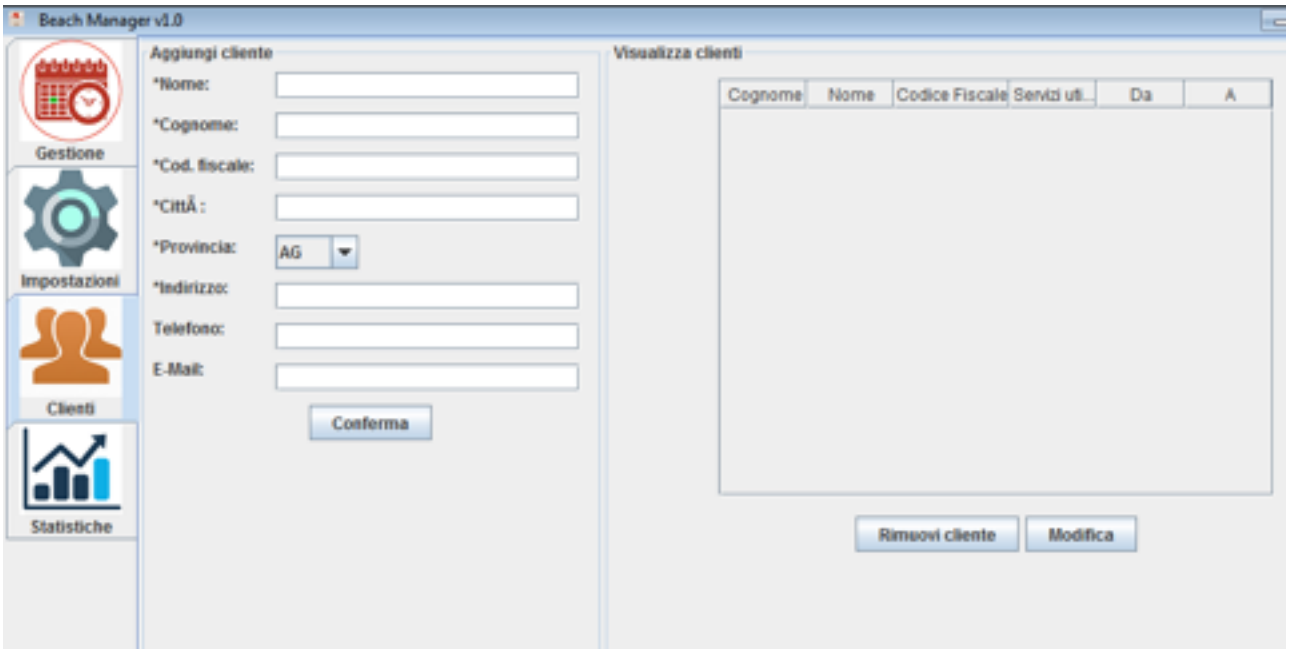
A questo punto si aprirà un nuovo Frame contenente le date selezionate nell'interfaccia precedente, tramite le freccette associate ad ogni servizio si può aggiungere o diminuire il numero dei servizi, tramite la spunta "pagato" si indica se il cliente paga al momento, altrimenti pagherà successivamente. Subito sotto verrà chiesta la tipologia di cliente, se generico o registrato, se registrato dovrà essere selezionato da una tabella. Una volta compilati tutti i campi verrà abilitato il bottone prenota per concludere la prenotazione.

Se invece si vuole visualizzare il calendario di un determinato ombrellone, nel frame "Gestione" basterà cliccare sul bottone "Visualizza calendario".

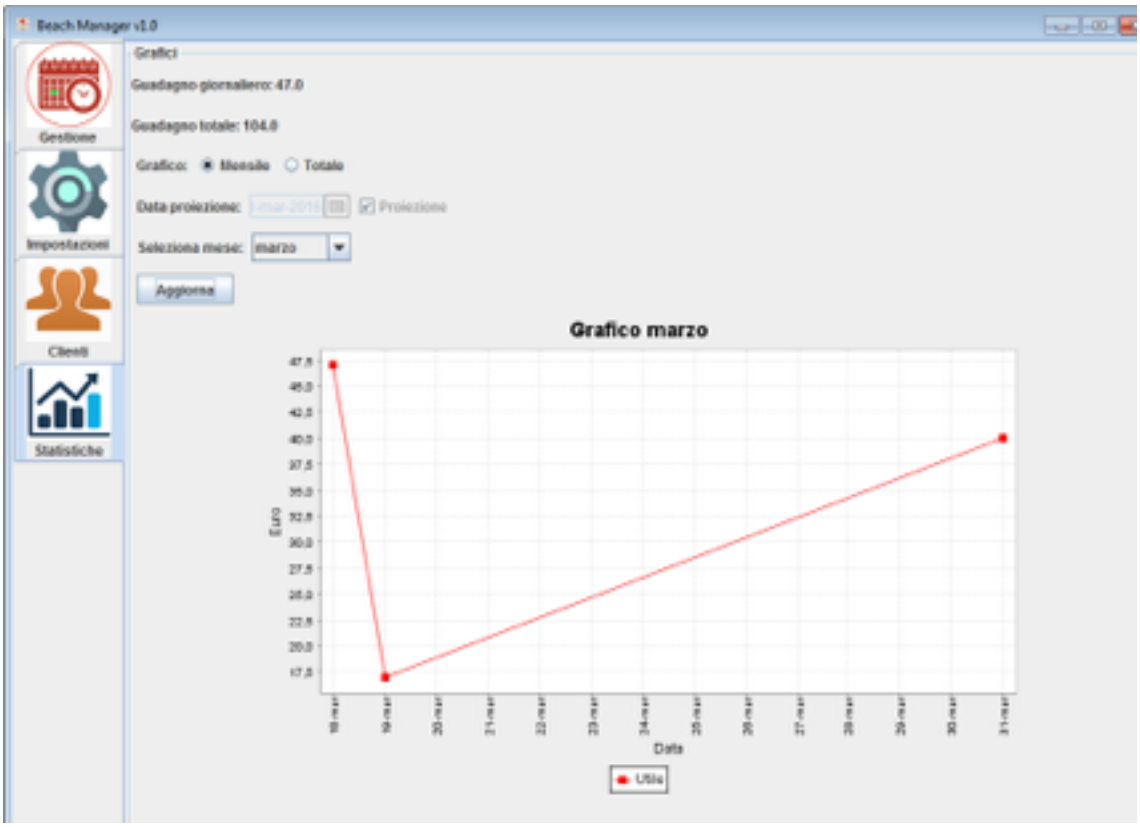


Dov'è possibile visualizzare o rimuovere una o più prenotazioni.

Cliccando nel tab a sinistra "clienti", si entra nella sezione dedicata, dov'è possibile aggiungere, rimuovere o modificare clienti. I campi con l'asterisco sono obbligatori, tutti gli altri opzionali.



Cliccando invece sul tab “Statistiche” si visualizzerà questa interfaccia



Dov’è possibile ottenere un grafico in base al guadagno mensile o totale, se abilitato il totale, si potrà visualizzare una proiezione del guadagno ad un determinato giorno (ad esempio voglio sapere quanto avrò guadagnato a

fine mese basterà selezionare la data e spuntare “proiezione”). Se abilitato mensile verranno visualizzati i vari guadagni per ogni giorno in cui ci sono prenotazioni attive.