MechOffice

Francesco Buresta

29 Maggio 2015

Indice

1. Analisi 3
1.1 Requisiti..... 3
1.2 Problema..... 3
2. Design 4
2.1 Architettura..... 4
2.2 Design dettagliato..... 5
3. Sviluppo 10
3.1 Testing automatizzato..... 10
3.2 Divisione dei compiti e metodologia di lavoro..... 10
3.3 Note di sviluppo..... 10
4. Commenti finali 11
4.1 Conclusioni e lavori futuri..... 11
4.2 Difficoltà incontrate e commenti per i docenti..... 12
A Guida utente 13

Capitolo 1

Analisi

1.1 Requisiti

Il software *MechOffice* è un gestionale che permette ad un' officina di organizzare sia la parte dell'accettazione del cliente, sia la parte dell'intervento finale sul veicolo consegnato all'atto dell'accoglienza.

Nel caso specifico *MechOffice* viene commissionato per un insieme di officine affiliate con il marchio "Mech-Office" avente convenzioni con determinate case automobilistiche ed un magazzino centralizzato a cui ogni officina fa riferimento e prezzo di manodopera unico.

Il committente ha richiesto le seguenti funzionalità.

- Una scheda in cui sia possibile curare l'accettazione del cliente dall'inizio alla fine dell'intervento.
- Fornire al cliente la possibilità di richiedere preventivi.
- Una sezione in cui sia possibile validare i pagamenti per le prestazioni svolte che funga anche da fattura di carattere puramente informativo.
- Una sezione in cui sia possibile aggiungere dati anagrafici dei clienti e loro vetture.
- Una sezione in cui gestire i propri dipendenti e fornitori.

1.2 Problema

Il problema principale è fornire una base per la gestione degli interventi in un' officina di autoriparazioni a partire dal primo contatto con il cliente fino alla fine della prestazione ad esso connessa. Fondamentale per la tracciabilità di ciascun intervento.

L'applicazione deve garantire persistenza dei dati dando la possibilità all'utente di salvare o caricare gli stessi.

Infine, la parte dell'interazione con il software deve essere molto semplice perché qualsiasi utente in pochi passi deve poter effettuare qualsiasi tipo di operazione.

Capitolo 2

Design

2.1 Architettura

Il software è basato principalmente sul pattern *Model-View-Controller* la cui struttura separa il modello (contenente la logica applicativa in grado di fornire metodi per accedere/salvare dati utili all'applicazione) dal controller e dalla view, garantendo un'ottima manutenibilità del software nel qual caso sia necessario intervenire sul codice per aggiornamenti e/o modifiche. Di seguito si mostra un possibile esempio.

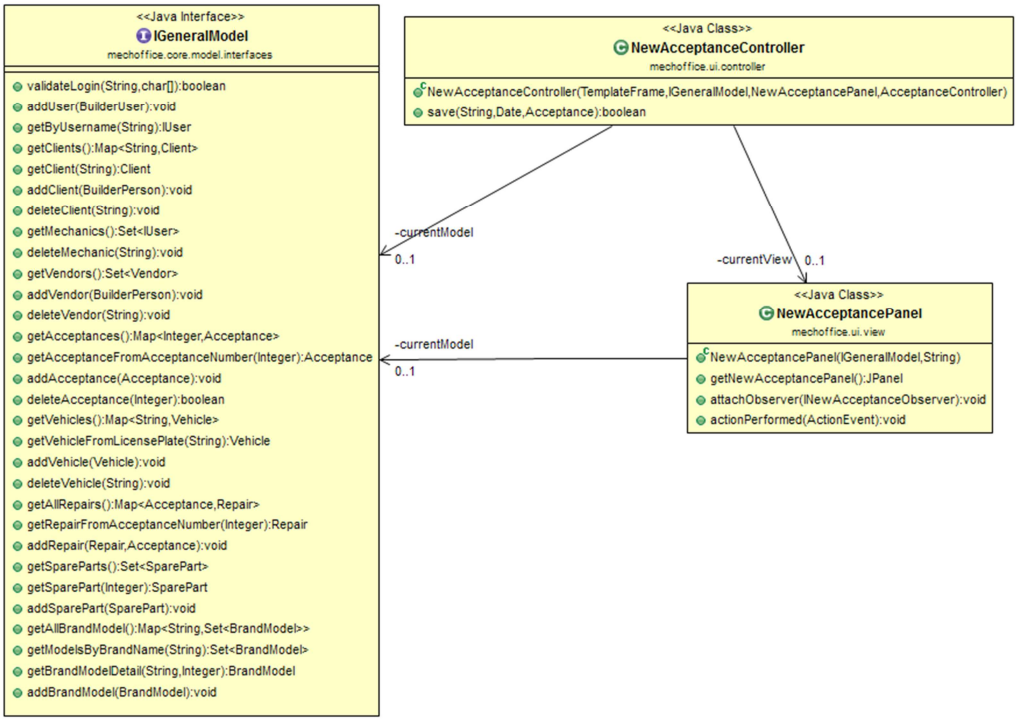

Figura 2.1.1: *Schema UML descrittivo dell'MVC accettazione del cliente con rappresentate le entità principali ed i rapporti fra loro.*

2.2 Design dettagliato

Di seguito verrà illustrata in modo dettagliato la struttura *core* del problema: Accettazione e Riparazione.

Accettazione

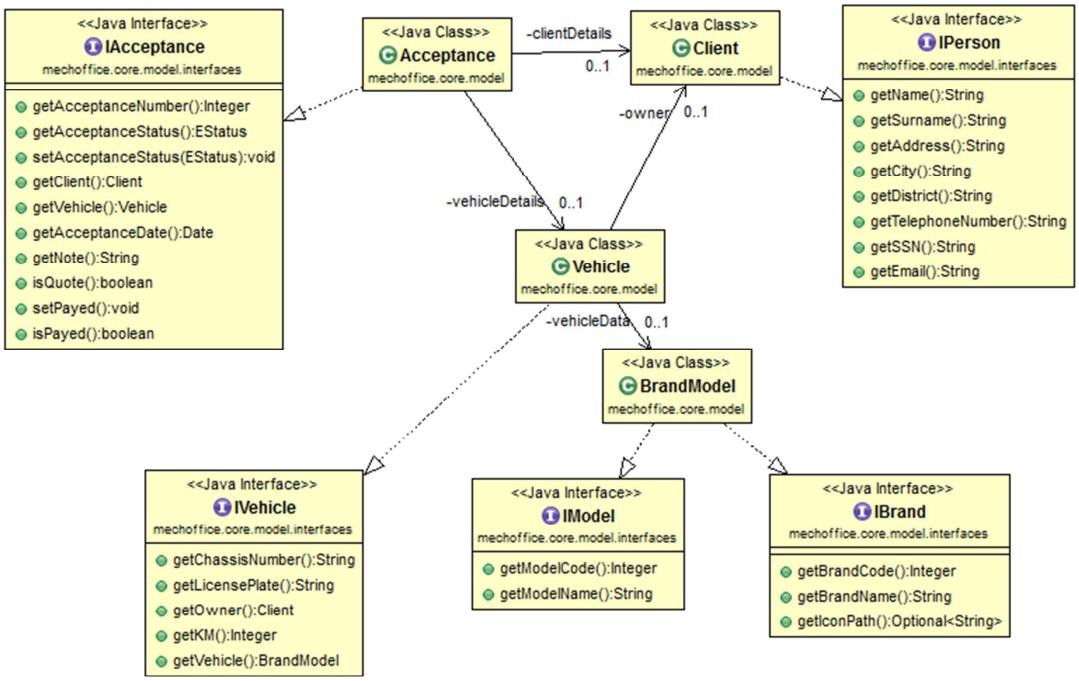

Figura 2.2.1: *Schema UML descrittivo dell'accettazione del cliente con rappresentate le entità principali ed i rapporti fra loro.*

Riparazione

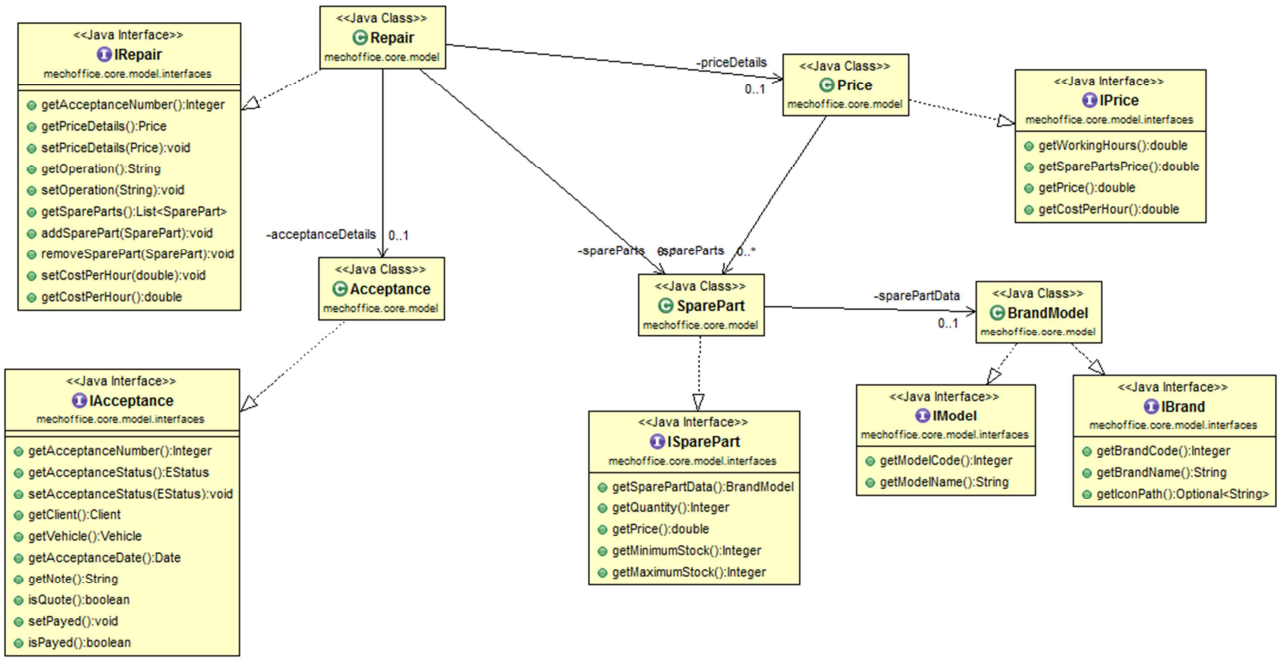

Figura 2.2.2: Schema UML descrittivo della riparazione per l'accettazione del cliente con rappresentate le entità principali ed i rapporti fra loro.

View

La view è stata realizzata rifacendosi al template *header, content, footer* utilizzato in ambito web. In particolare è stata realizzata una classe che implementa questo layout base ed inoltre viene suddiviso ulteriormente il *content* in *west,center,east* tramite una classe astratta che verrà estesa dalle view che personalizzeranno a loro volta il *content* secondo la struttura sopra descritta.

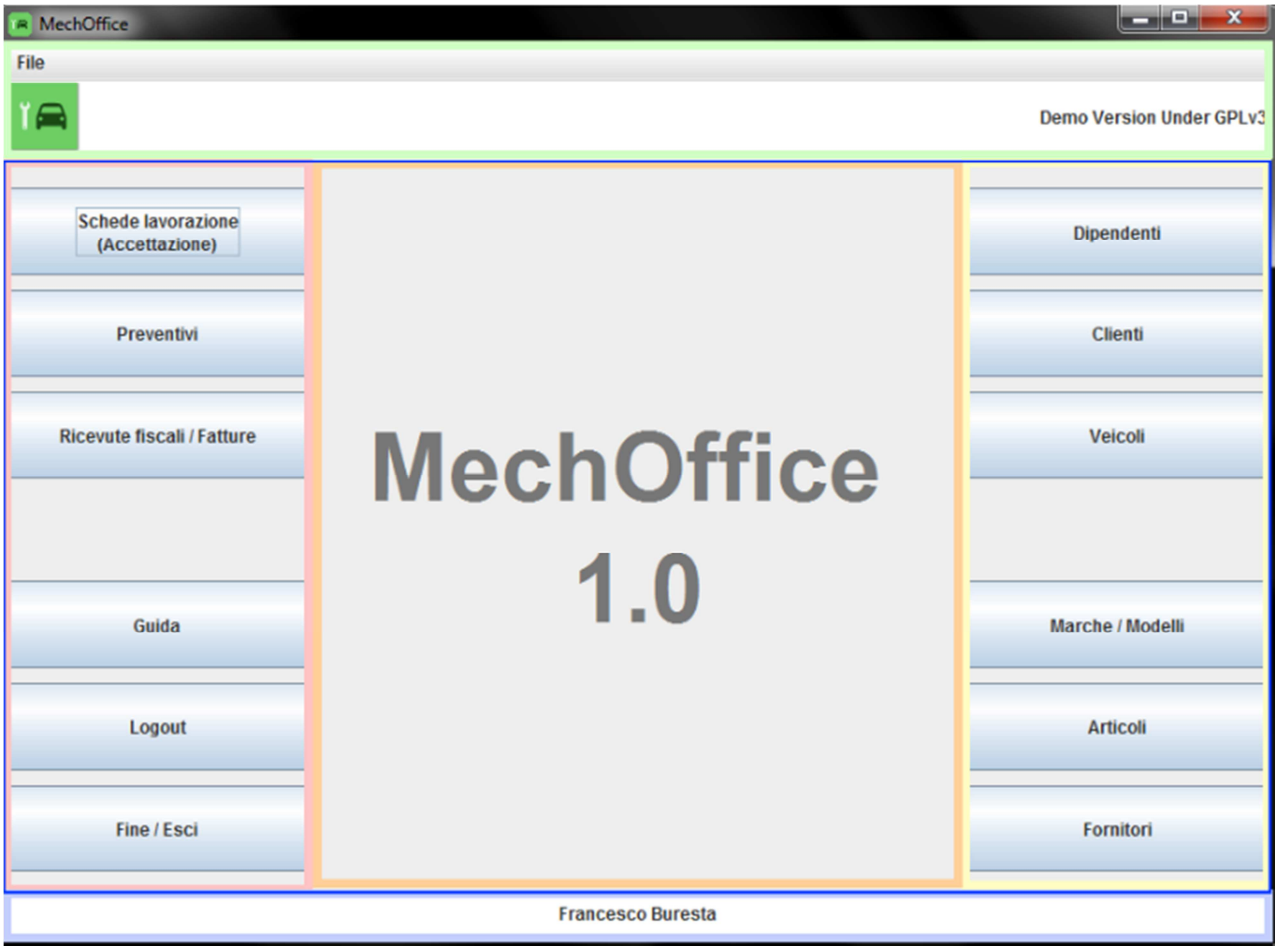

Figura 2.2.3 Divisione con template *header, content, footer* e ulteriore suddivisione del *content*.

In particolare :

- verde: *header*;
- blue *content*;
- light blue *footer*.

Suddivisione del *content* :

- rosa : *west*;
- arancio : *center*;
- giallo: *east;*

Pattern utilizzati

Nel software sono presenti altri pattern oltre al MVC:

Observer: Con questo pattern si vuol creare indipendenza tra le view e controller, ovvero, fare in modo che alla view non sia permesso di richiamare direttamente i metodi del controller, per questo motivo, la maggior parte dei controller sono legati alla rispettiva view per permetterne la gestione degli eventi ma la view scatenerà gli eventi su un'ulteriore classe observer la quale si occuperà di richiamare il giusto metodo del controller.

Questa struttura garantisce una flessibilità importante perché ogni view, fa riferimento all'observer e quindi se si interviene per un cambiamento sul controller, questo non causerà la modifica della view. Nel software viene usato diverse volte, in figura 2.2.4 si trova un esempio, ma il ragionamento viene iterato anche in tutti gli altri casi.

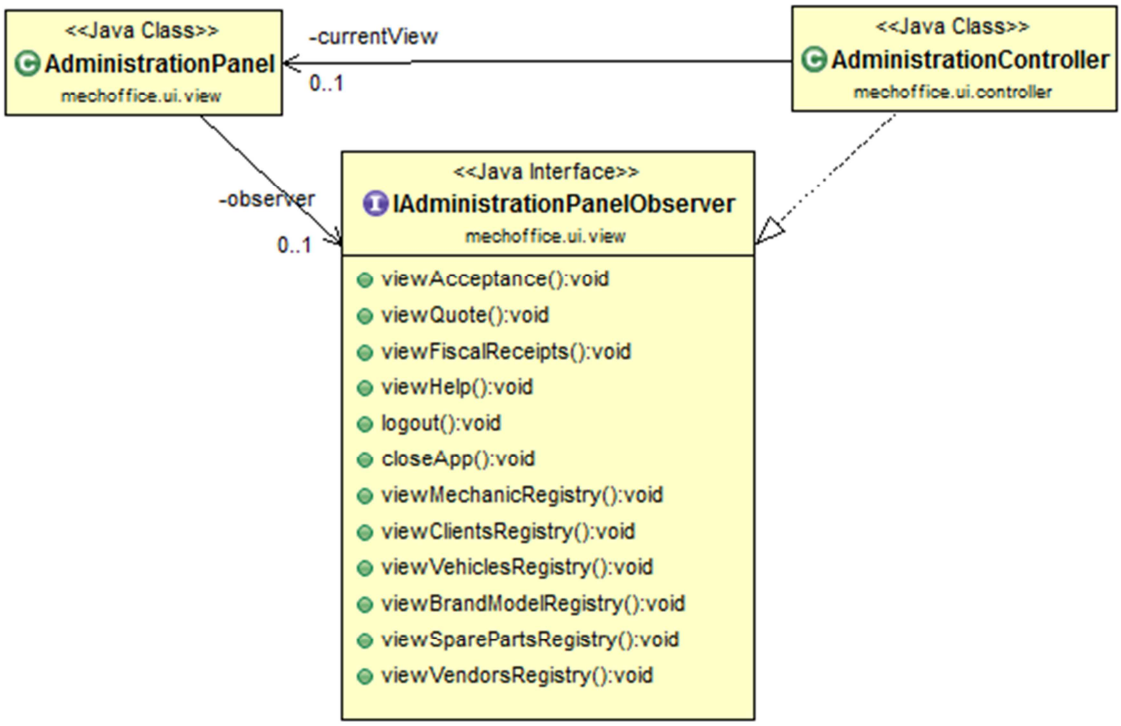

Figura 2.2.4 *Schema UML descrittivo degli observer dell'AdministrationPanel.*

Builder : Al fine di facilitare l'inizializzazione e la leggibilità, si è deciso di utilizzare questo pattern per quelle classi che presentavano un elevato numero di campi da inizializzare. In alcune classi è *nested*, per classi che descrivono una persona fisica, è dedicato (*BuilderPerson, BuilderUser)*.

Strategy : Per dinamizzare la creazione dei form di immissione dati (utente, persona o veicoli) o di pulsantiere, è stato utilizzato questo pattern che fornisce la possibilità di definire più varianti (*strategie*) di una famiglia di oggetti che variano indipendentemente dal loro contesto di utilizzo. Di seguito si riporta il caso del form di immissione dati per una persona fisica e la pulsantiera del pannello amministrativo (in cui sono stati omessi volutamente observer e interfacce del pannello per concentrarsi sullo Strategy).

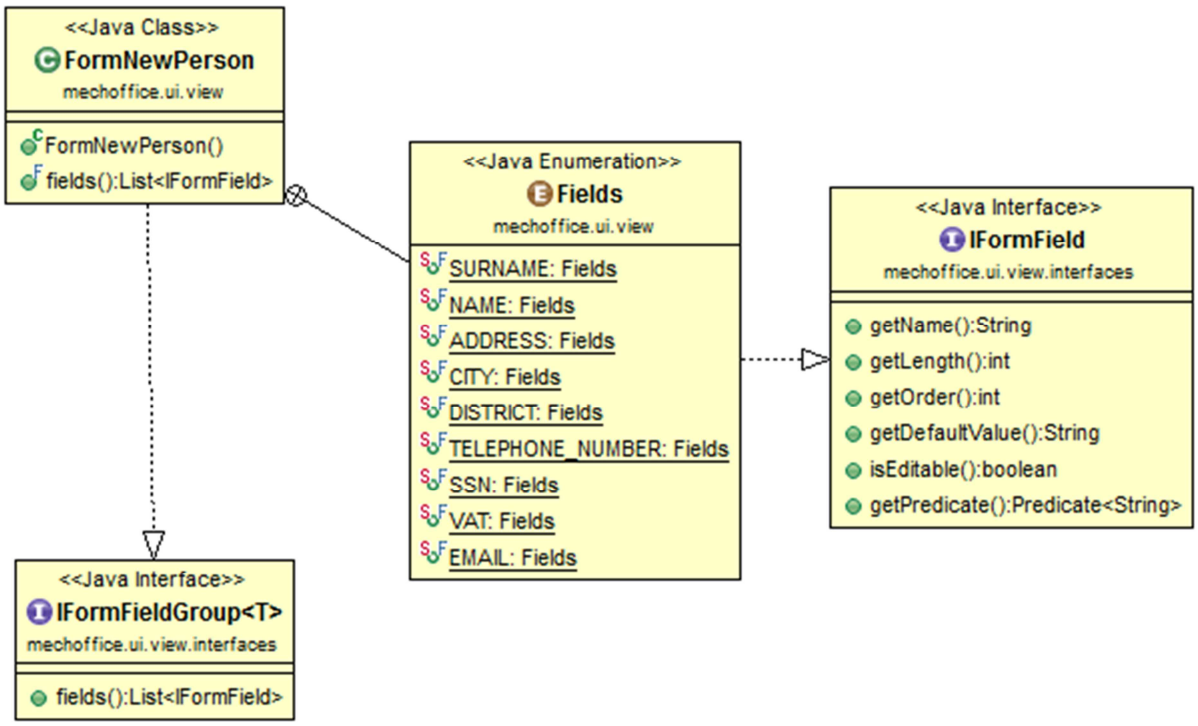

Figura 2.2.4 *Schema UML descrittivo dello Strategy per l'inserimento di una persona fisica.*

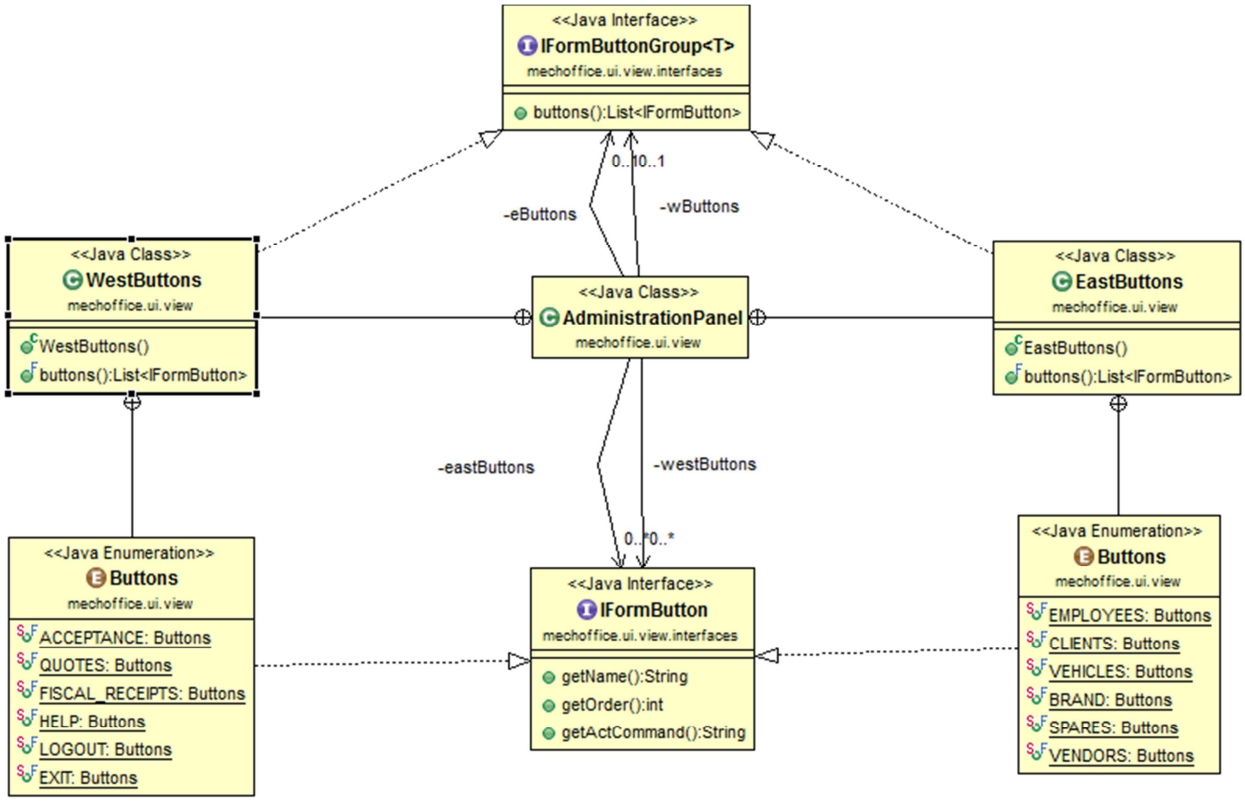

Figura 2.2.5 *Schema UML descrittivo dello Strategy per la pulsantiera del pannello amministrativo.*

Template Method : La classe AbstractCenterPanel implementa la divisione di un pannello in *west, center, east*, ovvero, il pannello è diviso verticalmente in tre parti: sinistra, centro, destra. Le classi che la estenderanno sceglieranno come utilizzare questa divisione (es. se utilizzare solo sinistra e centro). Le classi che utlizzano questo pattern sono : *RegistryPanel, MechanicPanel, AdministrationPanel* e *AcceptancePanel.*

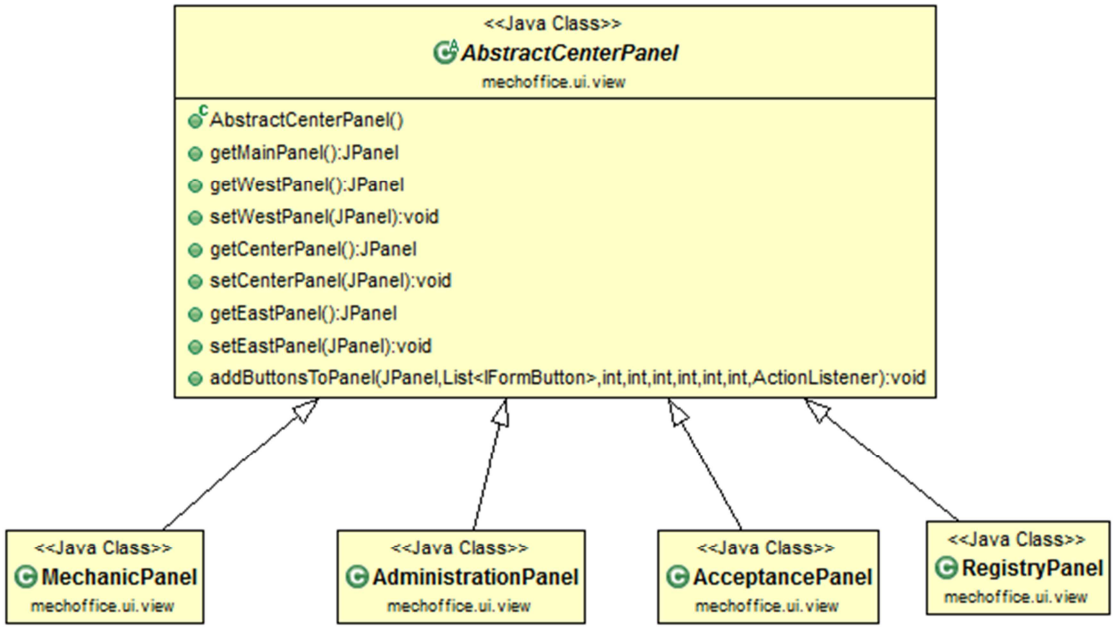

Figura 2.2.6 *Schema UML descrittivo del Template Method dell'AbstractCenterPanel.*

Capitolo 3

Sviluppo

3.1 Testing automatizzato

La parte del *model* è stata verificata con l'ausilio di test in JUnit, i quali hanno permesso di controllare non solo i casi in cui le varie funzioni avrebbero dovuto gestire i casi ottimi (ottenendo sempre in ingresso parametri corretti in base al dominio) ma anche quei casi in cui, in caso di immissione errata, avrebbero dovuto gestirne il problema restituendo un'eccezione.
Grazie a questo tipo di verifica sono state coperte tutte le casistiche d'utilizzo.
La parte *view* e relativi *controller*, invece, sono stati testati manualmente a causa dell'impossibilità di automatizzare un test del genere, poiché il funzionamento presuppone l'utilizzo da parte di un utente che ha interazioni con il software.

3.2 Divisione dei compiti e metodologia di lavoro

L'applicazione è stata sviluppata individualmente.

3.3 Note di sviluppo

Nell'applicazione sono state utilizzate le features di Java8, le cosiddette *Lambda Expression*; oltre a questo è stata impiegata una libreria esterna *SwingX* per i placeholder in alcuni campi di testo che rendono l'aspetto grafico più gradevole rispetto all'utilizzo di etichette vicino ai campi[1].
L'utilizzo del pattern *Strategy* è stato ripreso con modifiche dall'esame del corso di Programmazione ad Oggetti della Scuola di Ingegneria e Scienze Informatiche di Cesena a.a. 2014/2015 (nel dettaglio primo appello di Gennaio 2015, versione B, esercizio 2)[2].
Infine un *Timer* è stato prelevato con la sola modifica sulla durata da StackOverflow.com[3].

Capitolo 4

Commenti finali

4.1 Conclusioni e lavori futuri

Il progetto sviluppato per il corso di Programmazione ad Oggetti della Scuola di Ingegneria e Scienze Informatiche di Cesena è in una versione per così dire base-dimostrativa in cui si forniscono gli aspetti essenziali delle funzionalità che verranno ulteriormente revisionate e aggiornate nella versione completa.

Di seguito alcune features che si vorranno implementare e/o migliorare:

- realizzare un database decentralizzato per accedere ai dati
- implementare meccanismi di sicurezza sull'account utente
- semplificare la vista dell'accettazione garantendo una maggior velocità di inserimento dell'intervento.
- implementare grafici per statistiche di utilizzo e estrapolazione dei dati, con bilanci mensili e annuali
- implementare la fatturazione secondo le normative vigenti e il pagamento collegandosi a terminali esterni tramite porte RS232 o USB
- utilizzo di codici a barre per facilitare l'inserimento dei ricambi in fase di riparazione

Questa è una parte di quello che si intende svolgere per continuare a migliorare il software, l'obiettivo è poi quello di estendere la rete "MechOffice" a più officine sul territorio.

4.2 Difficoltà incontrare e commenti per i docenti

Durante la realizzazione del programma ci sono state leggere difficoltà dovute alla creazione delle viste senza l'ausilio di tool di progettazione ma questo non è stato altro che scuola poiché ho acquisito dimestichezza con alcuni layout messi a disposizione da Java.

L'utilizzo delle *Lambda Expression* ha permesso di snellire e rendere più comprensibili e compatte alcune parti di codice.

Inoltre ci sono stati problemi con l'installazione di Mercurial che mi hanno portato a trascurare l'importazione del progetto nei primi mesi.

Appendice A

Guida utente

Operazioni preliminari

Per poter eseguire l'applicazione dovreste aver installato correttamente *java 8* sul vostro sistema. L'avvio del software avviene semplicemente eseguendolo con la JVM (Java Virtual Machine – macchina virtuale Java).

Grazie all'interfaccia grafica semplice ed essenziale, in questa guida verranno riportate solamente le informazioni basilari di accesso al sistema e nozioni base dell'interfaccia.

Primo avvio

Dopo la schermata di presentazione di 4 secondi, viene mostrata una maschera di accesso.

Amministratore

È ora necessario fornire le credenziali dell'amministratore, l'amministratore è unico:

username : **adm01**

password : **12345678**

Validamento pagamento

Dalla schermata principale o dal pulsante "Schede lavorazione (accettazione), si ha accesso a tutte le schede di accettazione che sono state inserite, riparate e non pagate o preventivi non completi.
Per validare un pagamento occorre quindi cliccare due volta sulla scheda interessata (nella tabella) e nella finestra successiva presentare il conto dettagliato al cliente, in caso di pagamento, premere "Paga".

Nella sezione "pagamenti" a cui si ha accesso o tramite apposito pulsante "Pagamenti" o "Ricevute fiscali/fatture" (a seconda della schermata in cui ci si trova), sono elencati tutti i pagamenti che sono stati validati, per mostrare la relativa fattura di carattere puramente informativo cliccare due volta sul pagamento interessato.

Meccanico

Dopo esser stati registrati nell'applicazione dall'amministratore del software, effettuare il login con le proprie credenziali.

Nuova riparazione

Dalla schermata principale o dal pulsante "Schede lavorazione", si ha accesso a tutte le schede dell'accettazione che dovranno essere sottoposte a riparazione, o a preventivo. Per iniziare il lavoro, procedere al doppio click sull'accettazione interessata, la finestra appena aperta verrà utilizzata dal meccanico per la riparazione e conteggerà in automatico il tempo di lavoro una volta ultimata e quindi salvata.

Note generali per gli utenti

Dopo aver effettuato il login nel sistema, verrà presentata all'utente la schermata principale di MechOffice dalla quale è possibile accedere alle varie sezioni del programma.

In alto a sinistra è presente il menu *File* dal quale è possibile caricare una sessione (*carica*) o salvare (*salva*) quella corrente su un file con estensione .mms . Nella parte centrale invece è possibile selezionare l'azione desiderata cliccando sugli appositi pulsanti.

Quando si desidera eliminare una record dalle anagrafiche, cliccare sulla relativa riga con il tasto destro.

Si ricorda che la chiusura con salvataggio avviene solo tramite pulsante posto all'interno della finestra (*Fine/Esci*).

Bibliografia

1. http://java.net/downloads/swingx/releases/swingx-all-1.6.4.jar
2. https://bitbucket.org/mviroli/oop2014-esame/src/1769468b25124f00b6c905544936a6fb2e0d063e/ex2014/a01b/sol2/?at=default
3. http://stackoverflow.com/questions/6530157/javashow-message-dialog-for-10sec-and-remove