Relazione Progetto Domo

Corso di Programmazione ad oggetti 2014/15

De Mattia Simone
Falzaresi Stefano
Versari Marco

23 maggio 2015

Sommario

Il progetto è volto alla creazione di un sistema di gestione della domotica all'interno di un appartamento.

Lo scopo è di fornire all'utente un'interfaccia facile e semplificata che lo aiuti nella gestione di tutti gli apparati di sicurezza installati all'interno dell'abitazione.

Indice

Capitolo 1

Analisi

1.1 Requisiti

L'applicazione sviluppata dovrà permettere all'utente la creazione di un appartamento virtuale composto da stanze all'interno delle quali verranno inseriti dei dispositivi di svariate tipologie (ampliabili dinamicamente dall'utente).
L'attivazione di ogni dispositivo inserito potrà essere gestita sia in modo manuale che in modo automatico (attraverso delle schedulazioni).
L'utente sarà agevolato nella gestione da processi di salvataggio e ripristino automatizzati della configurazione creata.
Un'interfaccia grafica semplice e intuitiva agevolerà l'utente in tutti questi processi e permetterà di avere una visuale completa dello stato dell'intero appartamento.

1.2 Problema

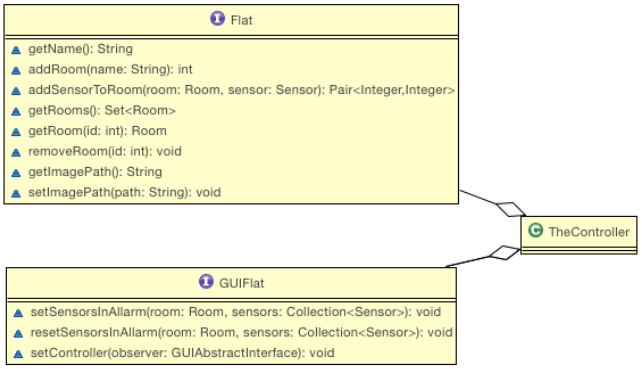
Le principali problematiche che dovranno essere affrontate sono quelle che riguardano la gestione "dinamica" dei dispositivi perché si vuol far sì che l'utente e i futuri implementatori possano gestire in modo autonomo il set di dispositivi da utilizzare all'interno dell'applicazione.
Questa dinamicità va quindi a impattare anche su tutte le procedure che gestiscono la grafica o il salvataggio e ripristino dei dati.
Per quanto riguarda la parte grafica, la problematica maggiore è quella di realizzare un'interfaccia che possa essere utilizzata in diverse tipologie di schermi senza perdere le posizioni e proporzioni dei dispositivi inseriti all'interno di un progetto.

Capitolo 2

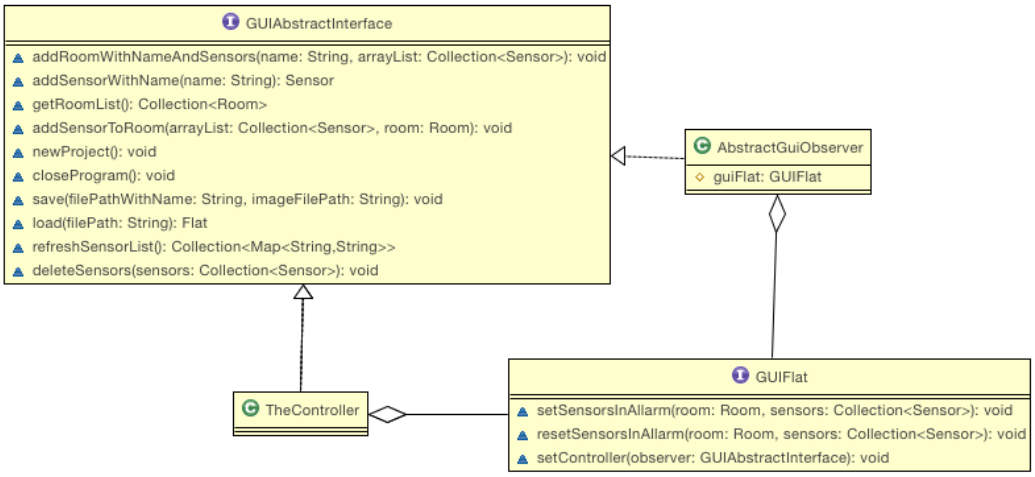
Design

2.1 Architettura

Principalmente si è deciso di utilizzare il pattern MVC (Model, View, Controller) in modo da poter generalizzare l'utilizzo di questo applicativo su dispositivi con diversi tipologie di interazione, grazie all'utilizzo di questo pattern infatti implementatori futuri potranno scegliere di riprogettare l'impatto grafico per permetterne il funzionamento su dispositivi embedded quali ad esempio Raspberry Pi semplicemente riprogrammando la parte View.



Si è scelto poi di utilizzare anche il Factory pattern che attraverso l'uso della Reflection ci permette di caricare le classi dei dispositivi in modo dinamico e anche tecniche come l'ereditarietà che hanno permesso di partire dalla progettazione di un oggetto generale per poi specializzarlo nel dettaglio.
Il pattern Observer è stato fondamentale nella comunicazione tra vista e controller e soprattutto per dividerli in maniera efficace, tanto da dover utilizzare solo la classe astratta fornita nel caso si voglia sostituire la parte di View.



Il diagramma mostra come il controller comunichi con la view attraverso i metodi esposti dall'interfaccia GUIFlat e come la view aggiorni il controller sulle operazioni fatte dall'utente attraverso la classe astratta.

La struttura generale del progetto è riassunta dal seguente diagramma:

dove a sinistra sono presenti le strutture dati: Flat, Room, Sensor. Al centro il controller e a destra la parte di view.

2.2 Design dettagliato

Package presenti:

- domo.bckrst: classi per il backup e il restore dei salvataggi con possibilita di criptare i dati.
- domo.devices: classi per la gestione dei sensori come classe astratta che invia messaggi al controller in caso di allerta.
- domo.devices.loader: classi dedicate alla reflaction e quindi al caricamento di sensori a tempo di esecuzione.
- domo.devices.sensors: classi di esempio, queste classi hanno la struttura corretta per poter essere caricate dal loader.
- domo.devices.util.counter: classi counter addette al'assegnazione degli id dei vari elementi.
- domo.general: contiene le classi generali come il controller o il main.
- domo.graphic: contiene le classi che costituiscono l'interfaccia grafica.
- domo.devices.test: test JUnit per le classi loader.
- domo.util.test: classi per il test dell'applicativo con interfaccia grafica.
- domo.educational.* : classi utilizzabili come laboratorio suddivise per argomento.

Oltre ai package elencati è anche presente la cartella "res" dove si trovano le immagini utilizzate per i pulsanti e anche per i sensori (nel caso in cui si voglia aggiungere un sensore sarà la cartella di destinazione dell'immagine assegnata a quel sensore).
Nel progetto è presente anche una cartella "classi" che è il percorso in cui andare aggiungere i nuovi sensori da caricare.

MODEL

La parte principale del model è strutturata da Flat, Room e Sensor dove l'oggetto flat è composto da un gruppo di room che sono a loro volta composte da una un insieme di sensori.

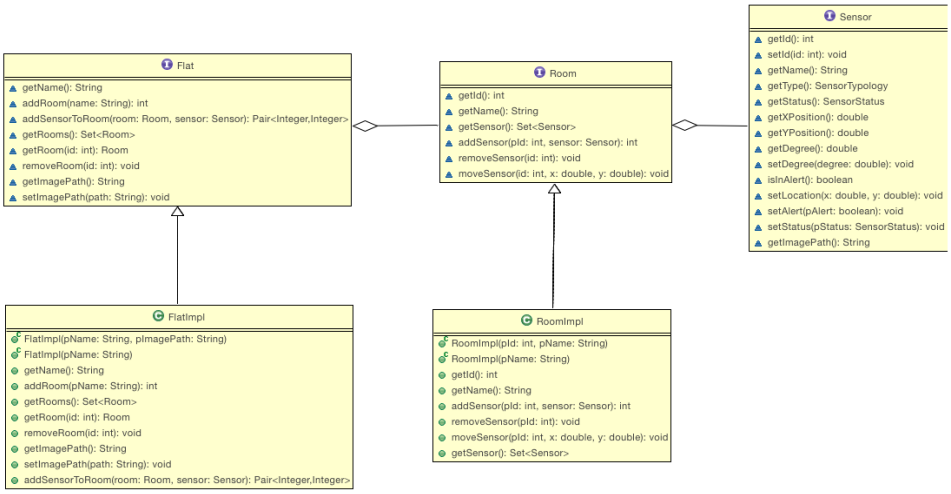


Figura 2.2.1 UML rappresentante la struttura generale del model

I sensori, grazie all'utilizzo della reflection, vengono caricati in modo dinamico runtime, l'interfaccia di un generico sensore "Sensor" è implementata all'interno della classe "AbstractSensor".

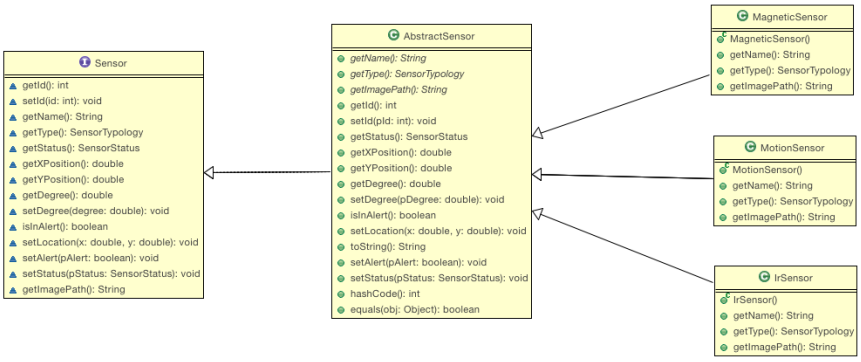


Figura 2.2.2 UML raffigurante l'implementazione del factory pattern applicato ai sensori

Chiunque intenda aumentare il set di sensori disponibili all'interno del progetto dovrà ereditare da quest'ultima classe e implementarne i metodi astratti.
La classe "Dinamic Loader" carica e controlla i nuovi moduli e verifica che le condizioni per il corretto funzionamento del progetto siano verificato (ad esempio viene verificato che esista un metodo "void setName(String x)" come da interfaccia).
È possibile verificare il corretto funzionamento di questo modulo attraverso la classe "WrongDevice".

Un'altra funzionalità del model riguarda le operazioni di backup e restore.
Il backup riceve dal controller l'oggetto flat di cui eseguire il salvataggio e attraverso l'utilizzo di oggetti xml salva tutti gli elementi in esso contenuti, il file di ripristino viene poi unito all'immagine dell'appartamento e salvato in un unico archivio.
La parte di restore effettua invece le operazioni opposte inserendo tutti i dati del progetto all'interno di una cartella Domo.
Per maggiore sicurezza sul file contententi le configurazioni viene effettuato una criptatura con algoritmo DES.

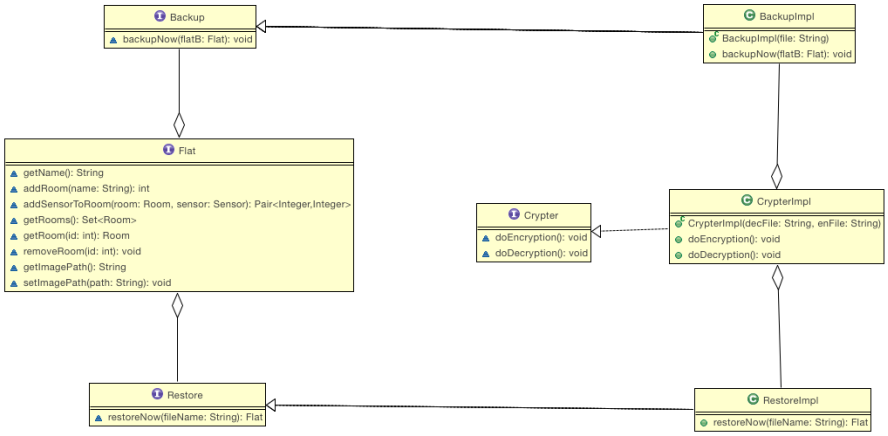


Figura 2.2.3 UML della struttura di Backup e Restore

VIEW

La view è composta da una classe principale GUIFlatImpl con interfaccia GUIFlat che estende la classe JFrame. Questa è la finestra principale che gestisce tutti i componenti di grafica.
Come layout manager si è scelto il BorderLayout per poter dividere le varie aree del frame in modo da avere in alto una serie di comandi principali, a sinistra un pannello che mostri lo stato dei vari sensori e la suddivisione in aree (room), in basso una barra dei suggerimenti e al centro l'area di lavoro dove l'utente crea il proprio progetto di domotica.

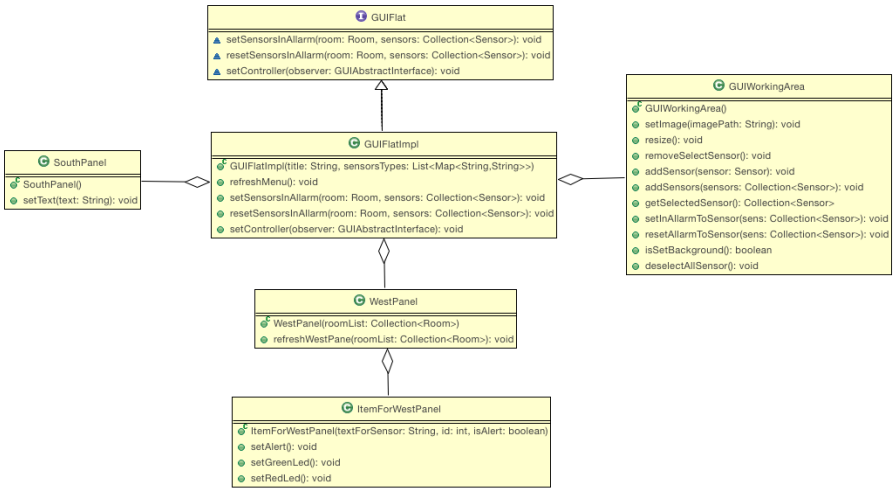


Figura 2.2.4 Schema UML del frame principale e delle classi di cui è composto.

La barra dei comandi oltre a mostrare comandi come "open", "save", "new" e i comandi utili per la creazione del progetto di domotica come ad esempio i pulsanti per poter aggiungere i sensori è anche dinamica per merito del caricamento di nuovi sensori a tempo di esecuzione. Quando si caricano nuovi sensori all'interno della cartella del programma basterà fare un refresh del menu per poter rendere questi sensori disponibili al progetto di domotica.
Per rappresentare graficamente un appartamento si è deciso di estendere una la classe JLabel e creare la classe ImageView. Per esempio è stato creato un metodo che rende l'immagine dell'appartamento sempre massimizzato in termini di dimensioni a seconda del JPanel in cui è contenuto. Infatti ha risolto il problema del ridimensionamento del frame principale. All'interno

sono presenti anche metodi per cambiare la scala dell'immagine e la sua rotazione.
Per la rappresentazione dei sensori si è fatta un'ulteriore estensione dalla classe ImageView creando la classe GUISensor. Questa classe ha metodi più specifici per la gestione dei sensori come l'implemetazione del MouseListener che permette la selezione, rotazione e spostamento dell'oggetto. In più è stato modificato il metodo setLocation(int,int) per far si che il sensore non possa essere posizionato all'esterno dell'immagine dell'appartamento.

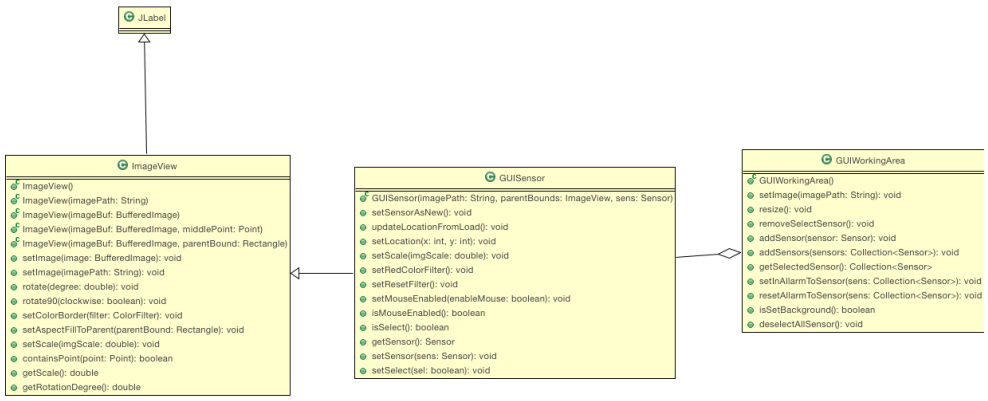


Figura 2.2.5 Estensioni della classe JLabel e il loro utilizzo all'interno della classe GUIWorkingAre

L’area di lavoro è gestita dalla classe GUIWorkingArea che è un’estensione della classe JLayeredPanel che da la possibilità di gestire a più layer gli oggetti al suo interno. Questa è la parte centrale del frame ed è l’effettiva area con cui l’utente interagisce per poter creare il progetto di domotica.

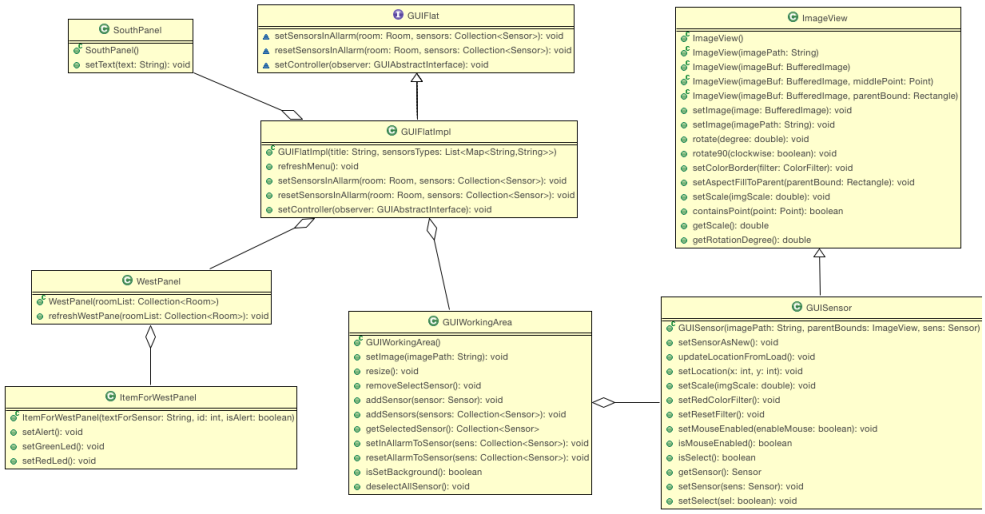


Figura 2.2.6 Schema generale della View

CONTROLLER

Il controller ha due principali compiti: raccogliere le richieste fatte dall’utente tramite l’interfaccia e ricevere le notifiche dei sensori quando questi sono in allarme e quindi aggiornare la vista.
Implementa GUIAbstractInterface, interfaccia della classe astratta AbstractGUIObserver che gli permette di ricevere messaggi provenienti dalla view.
In fase di test implementerà anche la classe AbstractTest (package domo.util.test) che permette di testare i sensori simulandone lo stato di allerta.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

L'applicazione è stata testata su:

- MAC OS X Yosemite 10.10.3.
- Windows 10 Insider Preview.
- Windows 8.1

Nel package domo.devices.test è presente il test JUnit per le classi Loader.

3.2 Divisione dei compiti e metodologia di lavoro

De Mattia Simone:

- Interfaccia grafica.
- Test grafico del funzionamento dei sensori.

Falzaresi Stefano:

- Backup e Restore.
- Sezione "Educational".

Versari Marco:

- Creazione delle classi del model come Sensor, Room, Flat.
- Reflection.

Parte in comune:

- Implementazione del controller.
- Debugging e risoluzione dei bug.

In comune è stata affrontata la progettazione dell'architettura di base, la dinamica di funzionamento da parte dell'utente finale e lo studio delle varie problematiche incontrate durante lo sviluppo. Una volta effettuata la progettazione iniziale ognuno dei membri ha sviluppato la propria parte in modo indipendente. Completate le parti indipendenti il gruppo si è concentrato nell'unione delle varie parti con la realizzazione del controller e infine per la fase di test nella ricerca di bug e nella loro correzione. Anche la risoluzione dei problemi che si sono incontrati durante lo sviluppo è stata affrontata in cooperazione.
La condivisione dei sorgenti prodotti dai membri del gruppo è stata fatta tramite Mercurial attravero il repository Bitbucket assegnando ad ogni membro una testa indipendente ed effettuando quotidianamente dei merge in modo da poter rimanere in linea con lo sviluppo effettuato dagli altri membri.

3.3 Note di sviluppo

L'interfaccia grafica è stata impegnativa, infatti, in via di sviluppo è stata riprogettata più di una volta e alla fine del progetto si è rivelata diversa dall'idea iniziale ma sicuramente più funzionale e intuitiva. On line sono state effettuate ricerche e trovate soluzione su come utilizzare l'oggetto BufferedImage per le funzionalità di scala, rotazione e per la copia di tale oggetto. Infatti viene utilizzata una copia da visualizzare dell'oggetto BufferedImage, perché si sono riscontrati problemi di definizione quando da una scala molto piccola si passava ad una grande (lo scale down faceva perdere definizione all'immagine).

Per quanto riguarda la parte di criptatura e decriptatura è stata fatta una ricerca sui principali siti di programmazione (Es. Stackoverflow codeproject) per definire il metodo pià adatto alle nostre esigenze

Capitolo 4

Commenti finali

4.1 Conclusioni e lavori futuri

Il progetto si è dimostrato abbastanza impegnativo e complesso anche per la necessità di un applicativo in cui gli utenti potessero creare, salvare e riprogettare la domotica di un appartamento.
Il progetto iniziale è stato rimodulato durante lo sviluppo per poter rientrare nelle tempistiche richieste dal docente.
Una delle parti che abbiamo dovuto eliminare è stata la gestione di uno scheduler che permettesse in modo automatizzato di attivare o disattivare i sensori, nel model è comunque presente una prima implementazione di questo comportamento.

4.2 Difficoltà incontrate e commenti per i docenti

Durante lo svolgimento del progetto abbiamo constatato che alcune parti come ad esempio la gestione di documenti xml sarebbero risultate interessanti se affrontate anche durante il corso.

Inoltre l'applicazione della reflection per il caricamento di classi di terze parti a tempo di esecuzione è risultata complessa e potrebbe richiedere un trattamento più approfondito.

4.2 Note di deployment

Durante i test di import progetto effettuati si è riscontrato un problema legato all'import da repository in un nuovo workspace di eclipse.

Il progetto viene correttamente importato ma non viene agganciato al workspace.

Per risolvere questa problematica una volta effettuato l'import da repository bitbucket premere nuovamente import ma selezionare "Existing Project into Workspace" ed indicare la cartella del workspace in cui è stato salvato il progetto.

La classe contenente il main risiede nel package domo\general e prende il nome di MainClass, eseguire la suddetta classe per avviare il progetto.

Appendice A

Guida utente

All'avvio del programma si sceglie se aprire un progetto precedentemente salvato o crearne uno nuovo (sia da barra dei pulsanti superiore sia da barra menu standard). Nel caso di nuovo progetto si dovrà scegliere un'immagine come planimetria di un appartamento.

Dopo aver creato il progetto si possono creare sensori nuovi semplicemente premendo il pulsante corrispondente al sensore che si vuole aggiungere. Quando i sensori sono contrassegnati da un bordo blu vuol dire che sono selezionati e in questo stato possono essere spostati (drag&drop), ruotati (tasto destro del mouse o rotella) oppure eliminati (pulsante cestino della barra dei pulsanti).

Per assegnare uno o più sensori ad una stanza (nuova o già esistente), si selezionano i sensori desiderati, si preme il pulsante di aggiunta nella stanza e si sceglie un nome in caso di stanza nuova o si sceglie una stanza dal menù a tendina, alla pressione del tasto "OK" i sensori verranno riassegnati alla stanza.

Il pannello a sinistra mostra la disposizione dei sensori e il loro stato (se in allerta oppure no).

Il progetto presenta anche una piccola sezione di test per dimostrare il funzionamento dell'applicazione. Consiste in un pannello con la lista dei sensori dove è possibile attivare o disattivare lo stato di allarme del sensore selzionato.