UNIVERSITÁ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA INFORMATICA ED ELETTRICA E MATEMATICA APPLICATA

Laurea Magistrale in Ingegneria Informatica

Relazione Project Work – Software Architecture & Design

Gruppo N°16 - AH

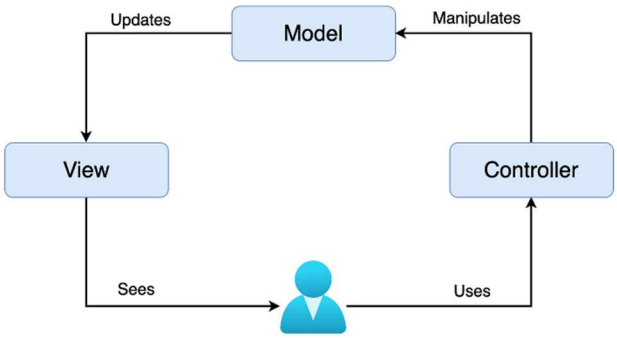
Studenti:

Cutolo Ciro	0622702532	c.cutolo7@studenti.unisa.it
Frasca Gaetano	0622702610	g.frasca10@studenti.unisa.it
D'Ambrosio Gennaro	0622702464	g.dambrosio65@studenti.unisa.it
Bottiglieri Alessio	0622702583	a.bottiglieri16@studenti.unisa.it

1 Description of software architecture

In questo paragrafo è descritta l'architettura adottata per lo sviluppo del software. Questa rappresenta la struttura fondamentale, il telaio, del sistema. Inoltre, definire un'architettura permette di stabilire i componenti principali, nonché le loro responsabilità.

1.1 Architettura



L'applicazione verrà costruita seguendo il paradigma architetturale **Model-View-Controller** (*MVC*), una scelta comune per lo sviluppo di applicazioni interattive, dotate di interfaccia grafica, come nel caso del software corrente.
Questa architettura permette una netta separazione della logica di dominio, contenuta nel Model, della visualizzazione grafica, contenuta nella View, e della gestione dell'interazione utente, contenuta nel Controller.

Formalmente:

1. **Model**: è responsabile della logica applicativa, gestisce i dati e garantisce la coerenza e l'integrità dei dati (non è a conoscenza né della *View* né del *Controller*).
2. **View**: è responsabile della rappresentazione grafica delle informazioni contenute nel *Model* e si occupa di fornire all'utente un'interfaccia intuitiva e reattiva, che si aggiorni in risposta alle modifiche del *Model*.
3. **Controller**: media tra *Model* e *View*, intercettando gli input dell'utente e interpretandoli và ad invocare le appropriate operazioni sul *Model*. Conseguentemente, aggiorna la *View* per riflettere le modifiche avvenute.

Tra i vantaggi principali di questo tipo di architettura abbiamo:

- **Manutenibilità e testabilità**: vi è una netta separazione tra logica applicativa e interfaccia, consentendo di testare e manutenere le singole componenti indipendentemente le une dalle altre.
- **Estendibilità e modularità**: l'architettura facilita l'aggiunta di nuove funzionalità, permettendo l'applicazione di modifiche localizzate alle componenti interessate senza impattare l'intero sistema, riducendo il rischio di introdurre regressioni nel sistema complessivo.
- **Collaborazione semplificata**: permette una suddivisione del lavoro più efficiente tra i membri del gruppo di lavoro, in modo chiaro. Ogni membro del team può concentrarsi su una componente specifica, in modo da evitare interferenze con il lavoro altrui.

1.2 Architecture and technology chosen

Data la scelta di MVC come archietettura abbiamo effettuato le seguenti scelte a livello di tecnologie:

- Linguaggio: Java
- GUI toolkit: JavaFX
- Testing: JUnit
- Versioning: Git + GitHub
- Gestione attività: Trello
- Development environment: NetBeans