Immagine che contiene scatola

Descrizione generata automaticamente

Università degli Studi di Salerno

Immagine che contiene testo, clipart

Descrizione generata automaticamente

Dipartimento di Ingegneria dell’Informazione ed Elettrica e Matematica Applicata

Corso di Laurea in Ingegneria Informatica

**Software Architecture Design 2024/2025**

**Canale I-Z**

**SOFTWARE ARCHITECTURE**

Gruppo n. **3 – IZ**

|  |  |  |
| --- | --- | --- |
| **Cognome e Nome** | **Matricola** | **e-mail** |
| Masturzo Vincenzo Pietro Pio | 0622702589 | v.masturzo@studenti.unisa.it |
| Santonicola Federico | 0622702567 | f.santonicola10@studenti.unisa.it |
| Sessa Attilio Marco | 0622702468 | a.sessa111@studenti.unisa.it |
| Treglia Martina Rosaria | 0622702656 | m.treglia3@studenti.unisa.it |

**Anno accademico 2024-2025**

# Description Of Software Architecture

Di seguito è riportata l’architettura del nostro sistema software, basata sul pattern Model-View-Controller (MVC):

Immagine che contiene testo, schermata, diagramma, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Figura 1: Architettura MVC

La scelta è ricaduta sull’ MVC in quanto è una architettura tipicamente utilizzata nel contesto di applicazioni utente che presentano un’interfaccia grafica, tramite la quale l’utente può interagire con il sistema e modificarne lo stato.

Questo modellodi architettura prevede la separazione delle responsabilità in tre componenti:

1. **Model:** gestisce tutte le forme geometriche, le loro proprietà (tipo, posizione, colore, dimensioni, rotazione, ecc.), lo stato dello spazio di lavoro, e la memoria delle operazioni.
2. **View:** responsabile della visualizzazione grafica del contenuto e dell’interazione con l’utente. Non contiene logica di business, ma solo presentazione.
3. **Controller:** gestisce gli input dell’utente (click, selezioni, comandi) e li traduce in azioni sul Model. Inoltre, aggiorna la View quando necessario.

L’adozione dell’MVC torna utile nel nostro contesto di sviluppo iterativo ed incrementale, in quanto permette di lavorare in parallelo su più componenti e favorisce il testing unitario e di integrazione.

Oltretutto, la separazione delle responsabilità in tre componenti ben distinti consente di isolare le modifiche a una singola parte del sistema, senza impattare direttamente le altre, migliorando così la manutenibilità, l’estendibilità e la riusabilità del codice.