**2. System design**

**2.1 Modello dell’architettura software**

**MVC:**

Il modello architetturale scelto è stato il pattern **MVC**, il quale utilizza un design pattern che suddivide il sistema in più moduli indipendenti e renderà più facili le modifiche e gli aggiornamenti.

Abbiamo usato il pattern **MVC** perché:

* È più facile riutilizzare il codice, quindi lo sviluppo è più veloce
* Il codice è più organizzato, quindi è più facile da capire e mantenere
* È più facile testare il codice.
* è più sicuro

Il pattern architetturale, è diviso in tre parti:

1. **Model**

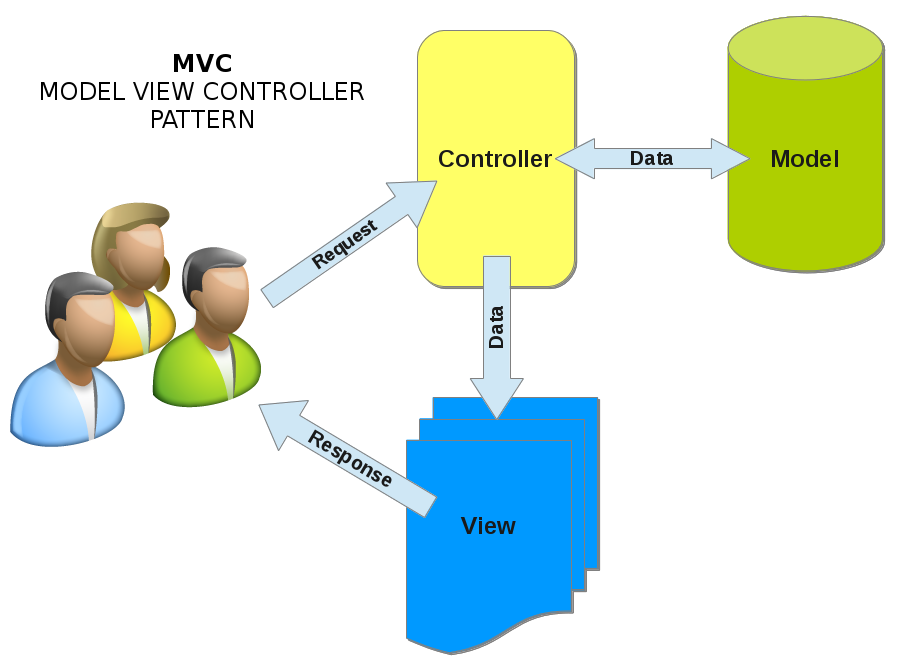
* Incapsula lo stato dell’applicazione
* Deve permettere di accedere ai dati
* Deve notificare i cambiamenti di stato

1. **View**

* Mostra il modello
* Gestisce l’interazione con l’utente

1. **Controller**

* Rappresenta la logica applicativa
* Collega le azioni dell’utente con modifiche dello stato
* Sceglie cosa essere mostrato



**DAO:**

Un altro pattern architetturale che è stato utilizzato, il DAO(Data Access Object) servirà per collegare un Database al sistema che gestisce tutti i dati degli utenti e delle opere nel sistema.

**VO:**

**2.2 Descrizione dell’architettura**

View:

Sarà composta da una schermata iniziale di login in cui verranno inseriti i dati di accesso e a seconda del suo ruolo avrà a disposizione delle operazioni. L’utente base ha a disposizione una barra di ricerca dove potrà inserire i dati per cercare l’opera nel catalogo e visualizzare quindi le opere. Gli altri utenti in aggiunta avranno a disposizione delle funzionalità aggiuntive come il download e la modifica di alcune parti dell’opera a seconda del proprio ruolo.

Model:

Gestisce le operazioni effettuate da GUI(Graphic User Interface) attraverso varie funzioni corrispondenti a ogni parte dell’interfaccia. Tramite questo modulo l’amministratore può modificare il ruolo di ogni utente,

Control:

In base ai dati di accesso dell’utente garantirà l’uso di determinate funzioni che potrà utilizzare