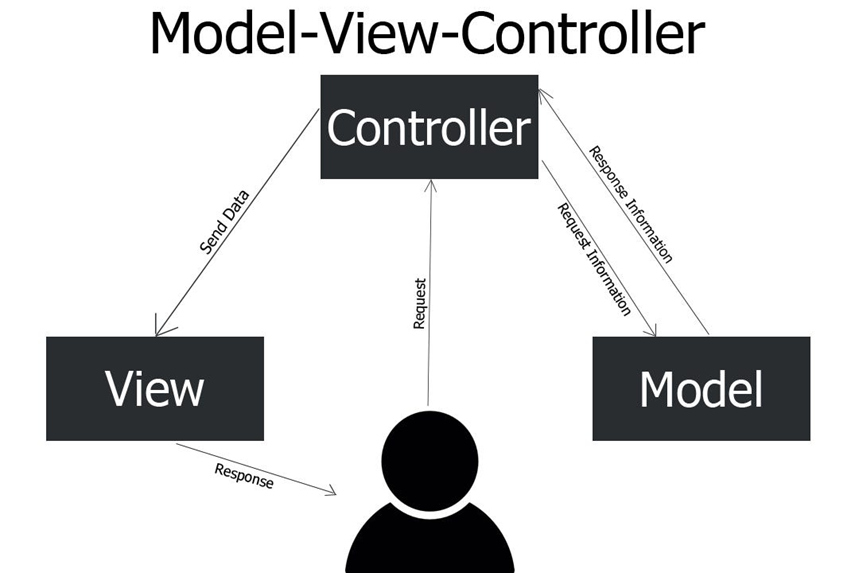
***Software Architecture***

Il software progettato si basa su un’architettura *Client-Server,* a sua volta implementata con uno stile architetturale *Model-View-Controller* (MVC):

* **Controller:** gestito dal *backend*, si occupa di astrarre e manipolare il contenuto dei dati del software, oltre a verificare che ogni funzione sia eseguita nel modo corretto; risponde alle esigenze del *client*, convertendone gli input in comandi per il modello e/o le viste*.*
* **Model:** componente centrale sempre gestito dal *backend*, permette un confronto con il modello preposto al funzionamento dell’intero sistema: lo scopo è creare un’applicazione che risponda a tutti i requisiti di partenza definiti, in termini di dominio del problema; risponde direttamente al *controller*.
* **View:** gestita dalla parte *frontend,* mostra la visione del software dal punto di vista del *client* (medico).



Nell’architettura MVC, l’interfaccia utente è separata dall’applicazione, proprio perché non è facile prevedere gli adattamenti richiesti dall’utente. L’obiettivo è creare un’architettura a servizi, che gestisca le varie richieste (dall’iscrizione al sistema dell’Ospedale fino al verbale medico dell’intervento), il tutto gestito da un solo e unico server; è possibile, tuttavia, che in futuro le varie parti del progetto possano essere utilizzate singolarmente, per rispondere ad uno dei vari servizi proposti dal sistema.

I componenti e connettori del progetto sono legati da *Dati condivisi*: poiché bisogna gestire una grande quantità di dati (tra informazioni dei pazienti e degli interventi), è necessario fare in modo che l’acceso a queste banche dati sia veloce e non confusionario.

***Componenti****: Memoria*, per ogni utente, che sia medico o paziente, bisogna mantenere una raccolta dati persistenti.

**Connettori:** *Chiamate di Procedura,* il controllo viene trasferito da un componente all’altro, anche attraverso procedure remote.

**Decisioni:** *Esplicite e Non Documentate*, per non creare ripetizione dei passaggi precedenti e per fornire un contesto e uno sfondo all’architettura software: vengono definiti i problemi di progettazione affrontati, lo stato del progetto, i presupposti ed eventuali alternative.

**Viste Architettoniche**

Per specificare l’architettura software è necessario usufruire delle cosiddette “viste”: esse permettono una rappresentazione di struttura e si dividono a seconda della componente analizzata.

* *Viste del Modulo*

Garantiscono una vista statica del sistema e nel nostro progetto sono stati sfruttati Diagrammi UML, per descrivere al meglio il funzionamento del progetto, anche attraverso l’uso di generalizzazione di classi (alcune classi ereditano attributi e operazione da un’altra classa, es. personale sanitario -> medici, infermieri e cabina di regia).

* *Viste dei Componenti e dei Connettori*

Forniscono un punto di vista dinamico del problema, descrivendo il sistema in funzione e, come già citato, nel nostro problema si è fatto uso di Dati condivisi.

* *Viste di Allocazione*Permettono di definire una relazione tra il sistema e il suo ambiente: bisogna intuire quali attori operano nelle varie parti del sistema. Si sfruttano Implementazione (per comprendere come il software viene gestito sulle strutture file) e si assegnano incarichi di lavoro ai diversi attori/classi, per ottimizzare il lavoro finale e non creare conflitti.