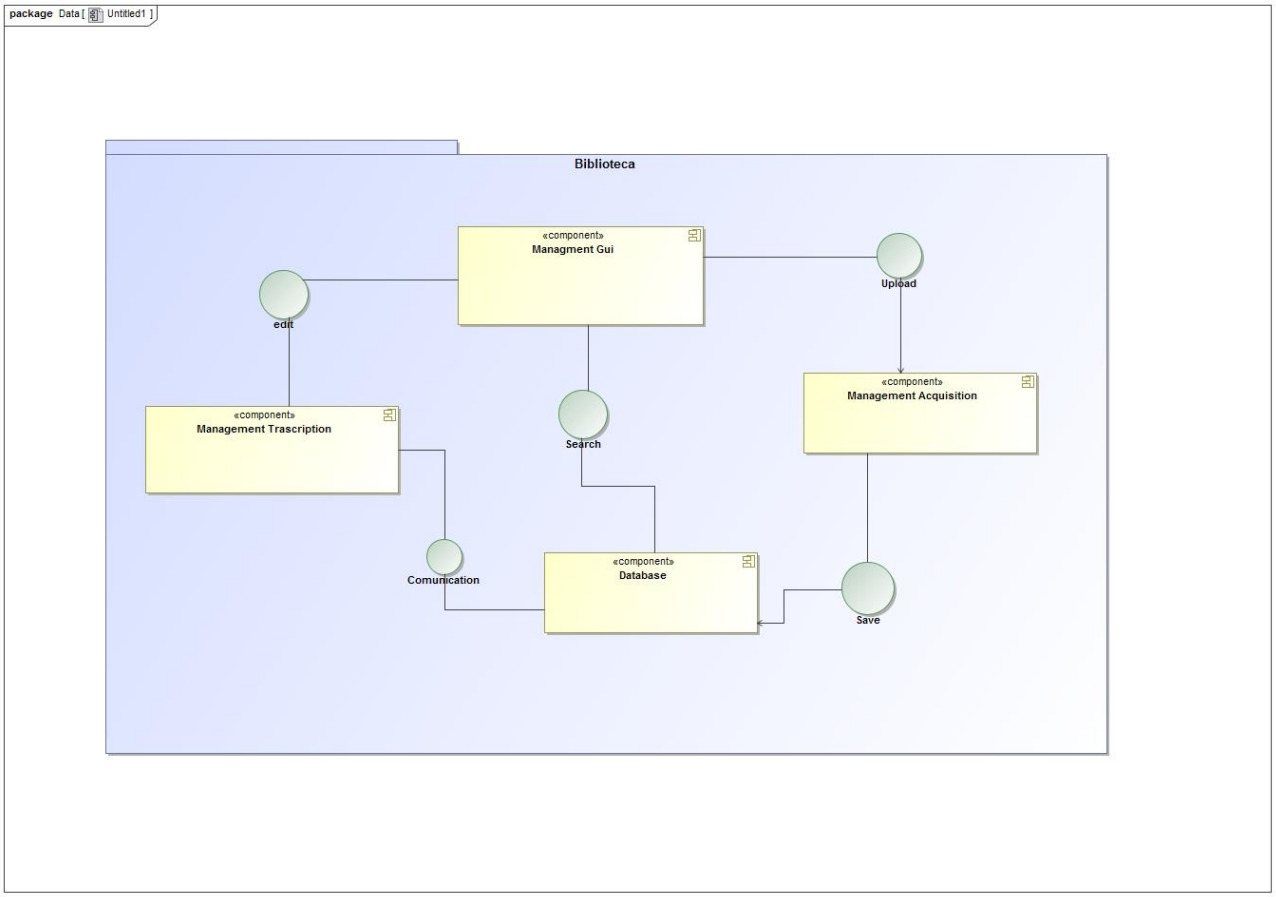
Modello dell’architettura software



Il Component Diagram in figura, ha lo scopo principale di mostrare le relazioni strutturali tra le componenti di un sistema (una componente è un elemento fisico rimpiazzabile del sistema).

Le componenti coinvolte sono le seguenti:

Management Gui : Rappresenta le interfacce grafiche con le quali l'utente utilizza il sistema. Si interfaccia con il database Attraverso l’interfaccia “Search” passandogli il nome del libro il quale poi verrà mostrato all’utente che ne fa richiesta.

Database: Questa componente rappresenta il database che conterrà le tabelle contenenti i “libri” e le varie informazioni degli utenti.

Management Acquisition : Rappresenta la componente che permette la scannerizzazione del documento. Si interfaccia con Management Gui attraverso l’interfaccia “Upload”, con la quale l’utente acquisitore avvia la componente e successivamente si interfaccia con il Database attraverso “Save”, passandogli l’immagine del libro la quale verrà salvata nel database.

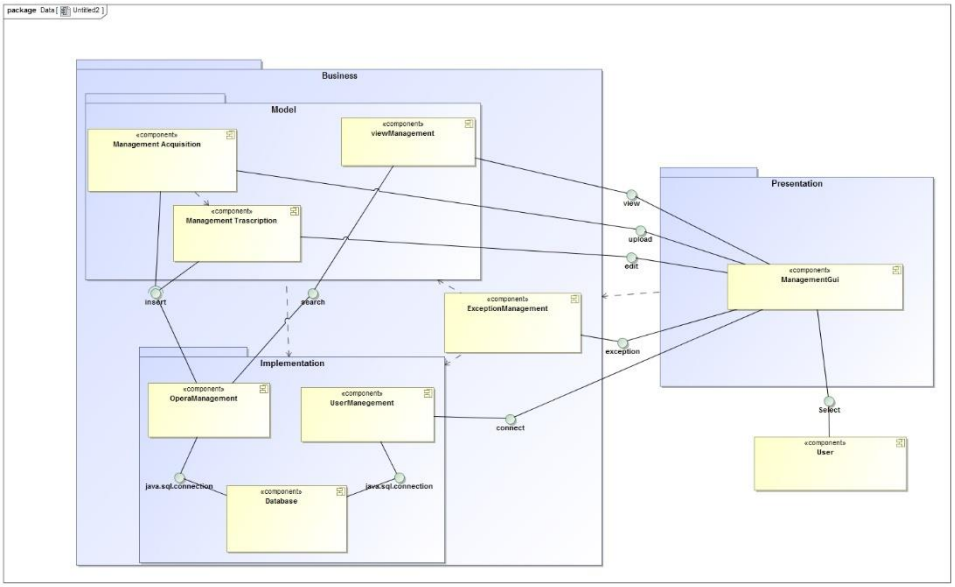
Management Trascription : Rappresenta la componente che permette al Trascrittore di trascrivere il libro. Si interfaccia con il database attraverso “Comunication”, per richiedere l’immagine di un libro. Inoltre si interfaccia con Management Gui attraverso “Edit”, con il quale il Trascrittore invia la richiesta per avviare il text editor TEI.

ARCHITETTURA CLIENT-SERVER

Abbiamo scelto questa architettura in quanto, i sistemi client/server sono un'evoluzione dei sistemi basati sulla condivisione semplice delle risorse (nel nostro caso libri): la presenza di un server permette ad un certo numero di client di condividerne le risorse, lasciando che sia il server a gestire gli accessi alle risorse per evitare conflitti.

DESCRIZIONE DELL'ARCHITETTURA SCELTA

(Client-Server)



CLIENT: PRESENTATION

SERVER: BUSINESS

Chi sono i Client e cosa fanno: Il software client in questo caso, è di limitata complessità, limitandosi normalmente ad operare come interfaccia verso il server. I client accedono ai servizi o alle risorse di un'altra componente, detta *server*, attraverso un'interfaccia (Gui) con la quale effettuano trascrizioni e/o acquisizioni delle immagini dei manoscritti oppure visualizzano opere e/o titoli in base al tipo di utente.

Chi sono i Server e cosa fanno: Il software *server*, oltre alla gestione logica del sistema, deve implementare tutte le tecniche di gestione degli accessi, allocazione e rilascio delle risorse, condivisione e sicurezza dei dati o delle risorse. Nel nostro caso attribuiamo a esso la gestione del Database.

DESCRIZIONE SCELTE E STRATEGIE ADOTTATE

Per la costruzione del sistema useremo la **piattaforma software** Java, in quanto, codesta è stata testata, perfezionata, estesa e provata da una community dedicata di sviluppatori, architetti e utenti appassionati di Java. Java è stato progettato per consentire lo sviluppo di applicazioni per sistemi portatili con prestazioni elevate per la più ampia gamma di piattaforme di elaborazione possibile. Rendendo disponibili le applicazioni in ambienti eterogenei, le aziende sono in grado di fornire più servizi, aumentare la produttività, la comunicazione e la collaborazione degli utenti finali e di ridurre significativamente il costo della proprietà delle applicazioni aziendali e di quelle di consumo.

Per quanto riguarda il database useremo **Sqlite.** Si tratta di una database library concepita in linguaggio C con lo scopo di implementare un DBMS SQL da utilizzare all'interno di un'applicazione. SQLite è un software Open Source che può essere liberamente utilizzato e modificato. L'uso di **SQLite** presenta numerosi vantaggi; innanzitutto si tratta di una libreria che richiede pochissimo spazio nel disco rigido, all'incirca 250 KB e questa caratteristica contribuisce a rendere i processi ad essa correlati molto rapidi e poco dispendiosi in termini di risorse.

Un'altra caratteristica da segnalare è quella riferita all'aderenza delle transazioni di **SQLite** allo schema **ACID** (**A**tomicity, **C**onsistency, **I**solation and **D**urability), cioè:

1. **Atomicità**, per cui ogni transazione è indivisibile durante l'esecuzione;
2. **Consistenza**, per cui viene salvaguardata la consistenza del database all'inizio, durante e alla fine della transizione;
3. **Isolamento**, per cui ogni transazione viene eseguita isolatamente e indipendentemente dalle altre;
4. **Durabilità**, per cui le modifiche apportate divengono persistenti e non vengono perdute;

Per lavorare l'interfaccia grafica useremo SWING, che è una libreria di Java leggera e facilmente utilizzabile.