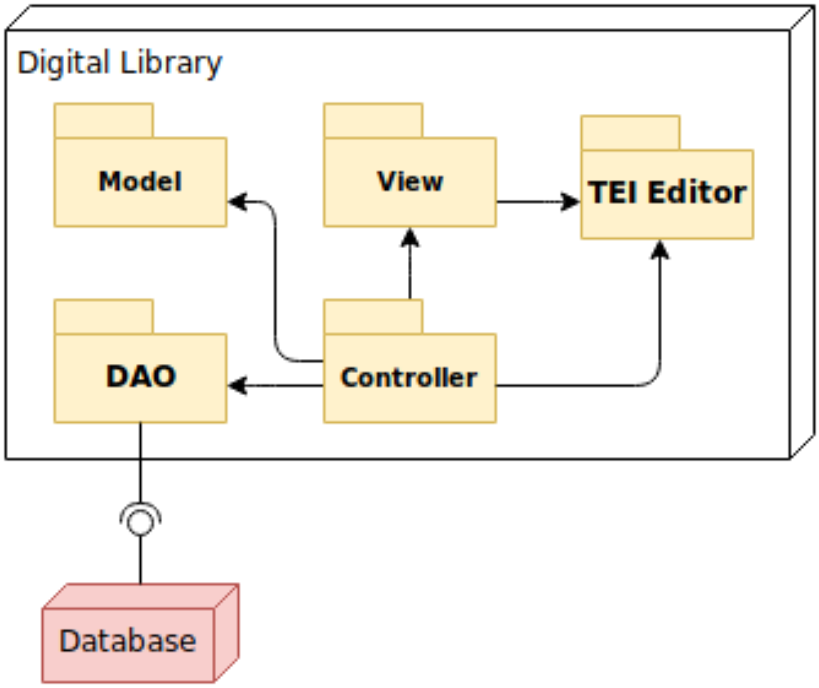
Descrizione dell’architettura

Il sistema è stato progettato tenendo a mente i requisiti di scalabilità, flessibilità ed estensibilità. La scelta è stata quella di seguire un pattern architetturale Model-View-Controller.

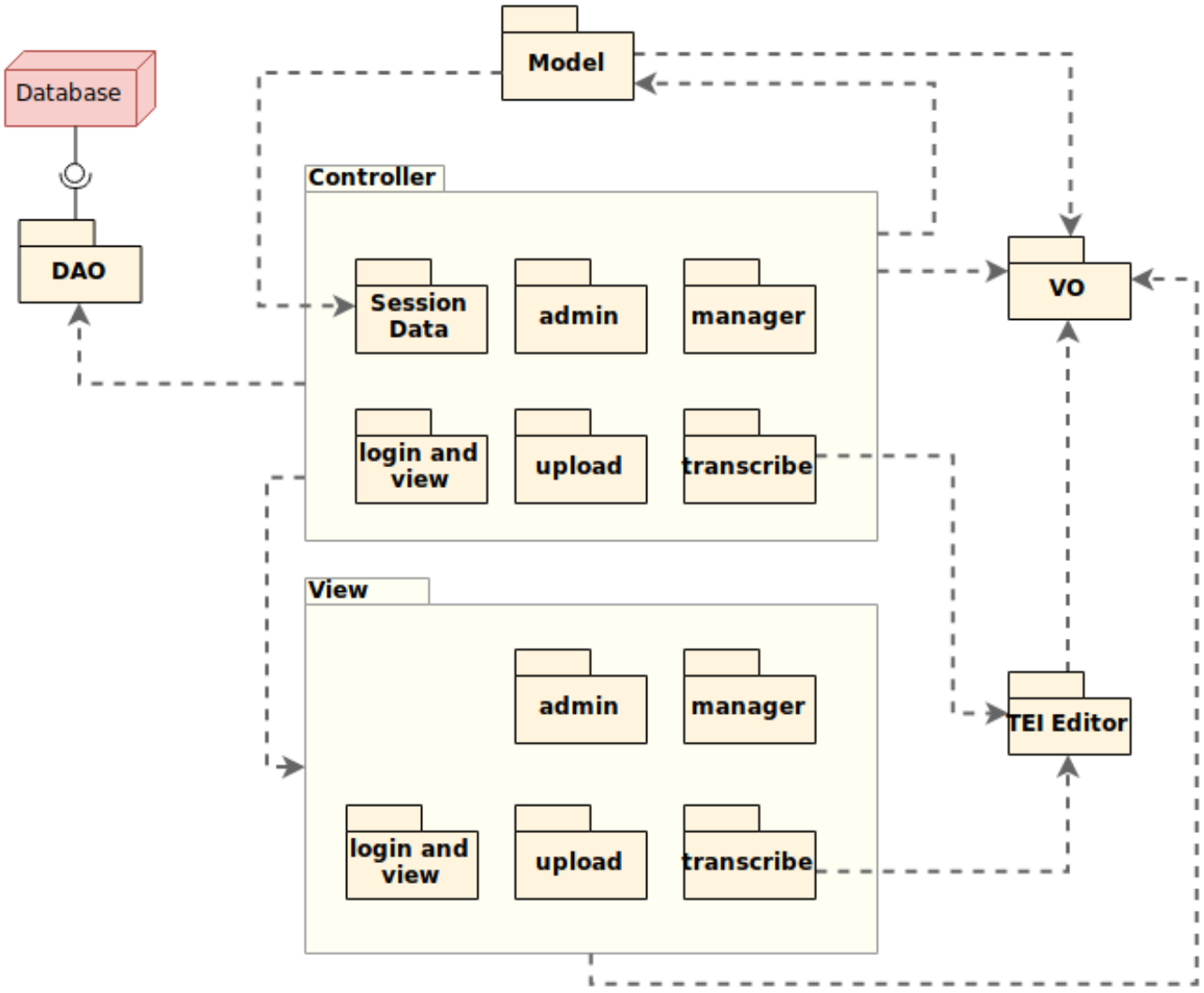
Oltre ai tre package **Model**, **View** e **Controller** sono presenti **VO**, **DAO** e **TEI Editor**; il package **VO** contiene semplici classi utilizzate frequentemente nel sistema, **DAO** si occupa di astrarre l’accesso al DBMS, **TEI Editor** sfrutta una libreria esterna per implementare l’editor TEI per le trascrizioni (quindi per ora non è stato considerato).

Si era pensato di aggiungere un package per la gestione dell’autenticazione e dei permessi degli utenti sfruttando delle librerie esterne ma si è ritenuto che a questo livello di progettazione avrebbe aggiunto complessità eccessiva al sistema.

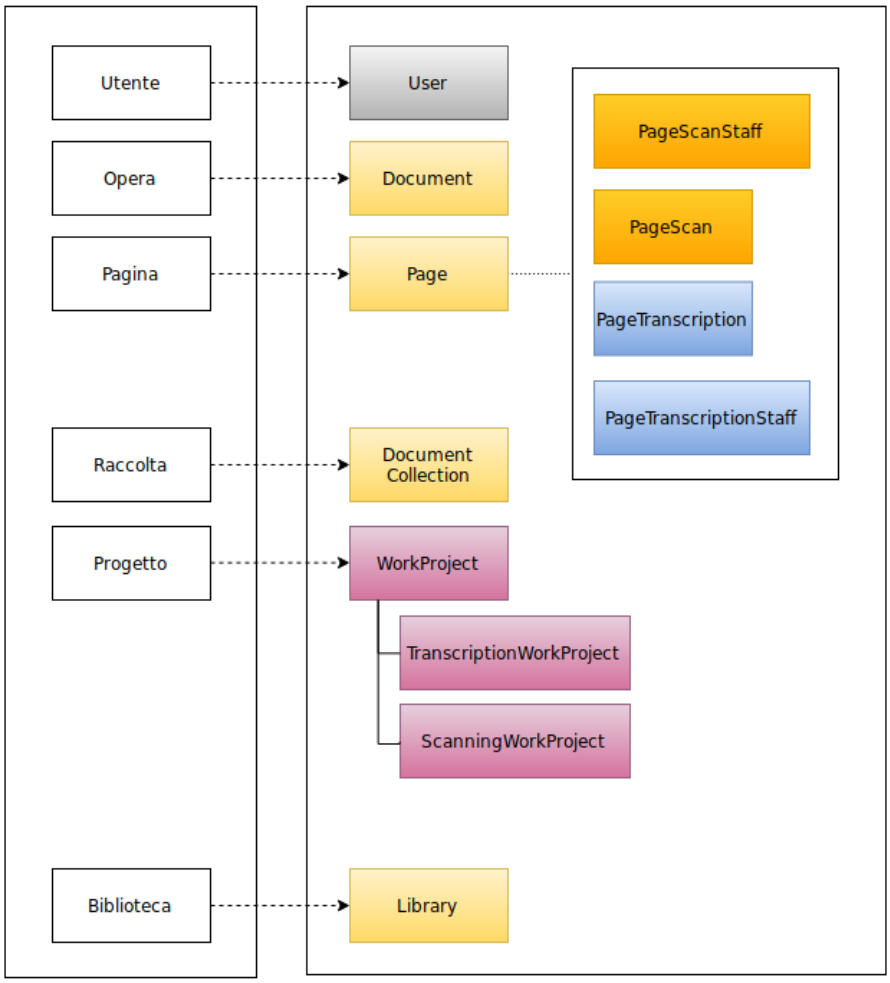


Schema dei package, omettendo il VO

Scendendo più nel dettaglio, si possono raggruppare le funzionalità di **Controller** e **View** per sottosistemi e farvi corrispondere dei subpackages. Le funzionalità di registrazione e autenticazione sono state accorpate al sottosistema Viewer. Nel package **Controller** è inoltre presente un subpackage *Session Data,*che permette l'astrazione del caricamento in memoria delle classi nel **Model**, in modo da non complicare eccessivamente la struttura del **Model** ma rendere il sistema scalabile.



Nel Model sono presenti le Classi derivanti dalle entità emerse nel modello di dominio, con alcune modifiche per evitare classi eccessivamente grandi, fornire una separazione tra parti logicamente distinte delle entità (nel caso delle pagine e dei progetti) e rendere più flessibile l'implementazione (anche in vista di modifiche future).



Schema delle conversione delle entità presenti nel modello di dominio nelle classi del package Model