MyTEILibrary

Sommario

1.1 Documento dei requisiti

Il sistema prevede diversi ruoli per ogni utente, ipotizziamo quindi l'uso di una maschera di login per tale scopo. In base ad ogni ruolo identificato verranno concesse o negate diverse funzionalità sulle varie maschere.

Le funzionalità principali saranno quindi quella di login, che in base all'utente autenticato costruirà il proprio menù di interesse in base al proprio ruolo

Avremo una serie di maschere per poter interagire con il sistema:

- Una maschera di **acquisizione** e una di **revisione e pubblicazione dei documenti acquisiti** (per motivi di semplicità abbiamo pensato ad una maschera di upload di scansioni in formato immagine, invece di acquisirle da scanner evitandone l'interfacciamento in questa fase).
  L'utente potrà caricare un'immagine se ha i diritti di acquisizione, inserire informazioni accessorie quali titolo ed autore dell'opera e una descrizione sull'immagine specifica acquisita.
  L'utente con i diritti di revisione acquisizione avrà invece la possibilità di richiamarla, visualizzarla, ingrandirla ed analizzarla nel dettaglio. Se è conforme potrà decidere se pubblicarla per essere visualizzata sulla maschera di trascrizione.
- Una maschera per **la trascrizione di documenti** che consenta di navigare liberamente l'immagine acquisita da un lato e che consenta la trascrizione in un formato quanto più simile al layout originale dall'altro, permettendo in un secondo momento la trasformazione in linguaggio TEI.
  L'utente con i diritti di trascrizione potrà richiamare un'immagine, caricata da lui o da altri utenti del sistema,acquisita e pubblicata dalla maschera precedente e trascrivere quello che vede dal pannello dell'immagine navigabile, sul pannello con l'editor di testo integrato.
- Una maschera di **revisione e pubblicazione del testo acquisito**, accessibile ai soli utenti abilitati.
  L'utente con diritti di revisione trascrizione, può richiamare una trascrizione di un utente (potranno esisterne più di una per la stessa acquisizione), revisionarla ed eventualmente pubblicarla.
- Una maschera di **visualizzazione titoli opere** a sistema e, qualora l'utente fosse abilitato, la possibilità di **leggere l'intera trascrizione** del documento.
  L'utente con i diritti base potrà solamente vedere una lista delle opere trascritte a sistema.
  L'utente con i diritti avanzati potrà invece, oltre che consultare i titoli delle opere, leggerne l'intera trascrizione pubblicata ed esportarne **il contenuto in formato TEI**, su un file di testo.

Il controllo abilitazione utente sarà duplice: la costruzione del menu avviene per ruolo collegato all'utente, per cui non sarà passibile lanciare una funzionalità non direttamente collegata al ruolo. Il secondo controllo avviene per abilitazione esplicita sul ruolo stesso: al momento di lanciare una funzionalità, verrà eseguito un controllo sulla tabella dei ruoli dal DB, per verificare che effettivamente la variabile di abilitazione funzionalità sia stata settata a true. In caso contrario, l'utente avrà un messaggio di avvertimento di funzionalità non inclusa nel suo profilo.

2.1 Modello dell'architettura.

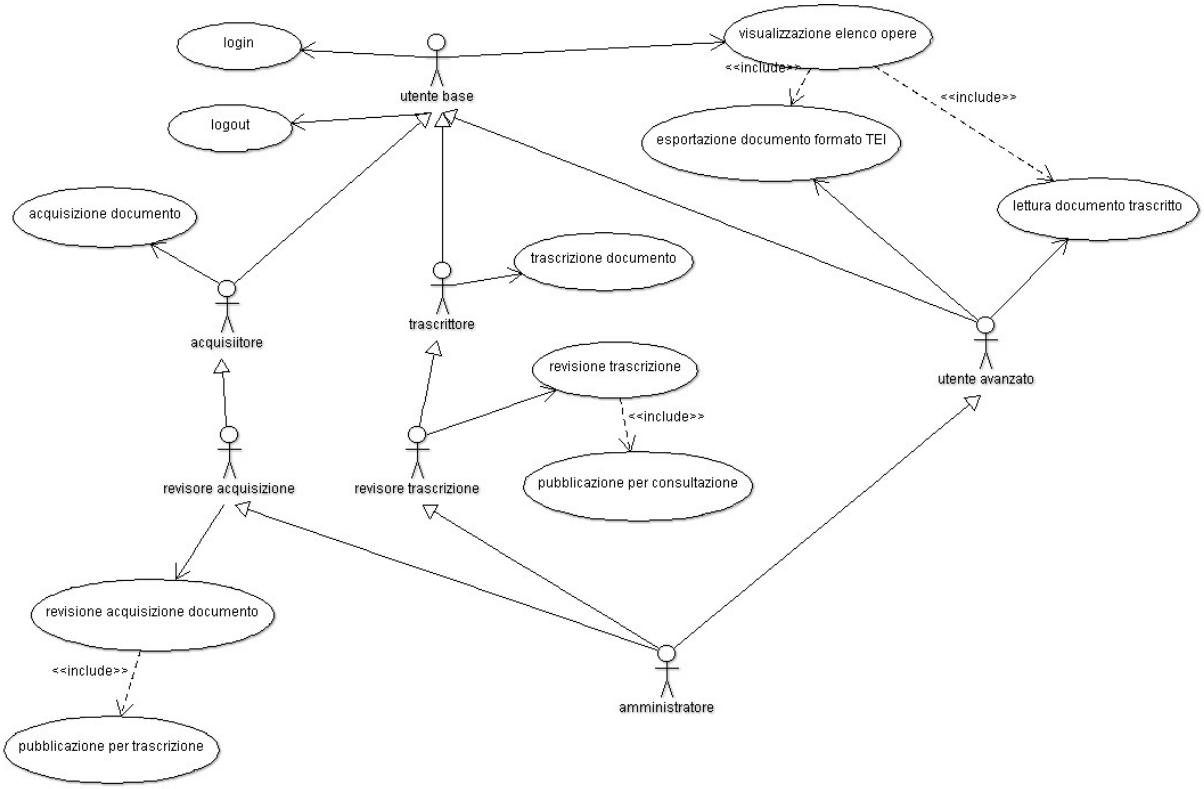


Figura 1 Use Case con utenti,relativi ruoli e funzionalità

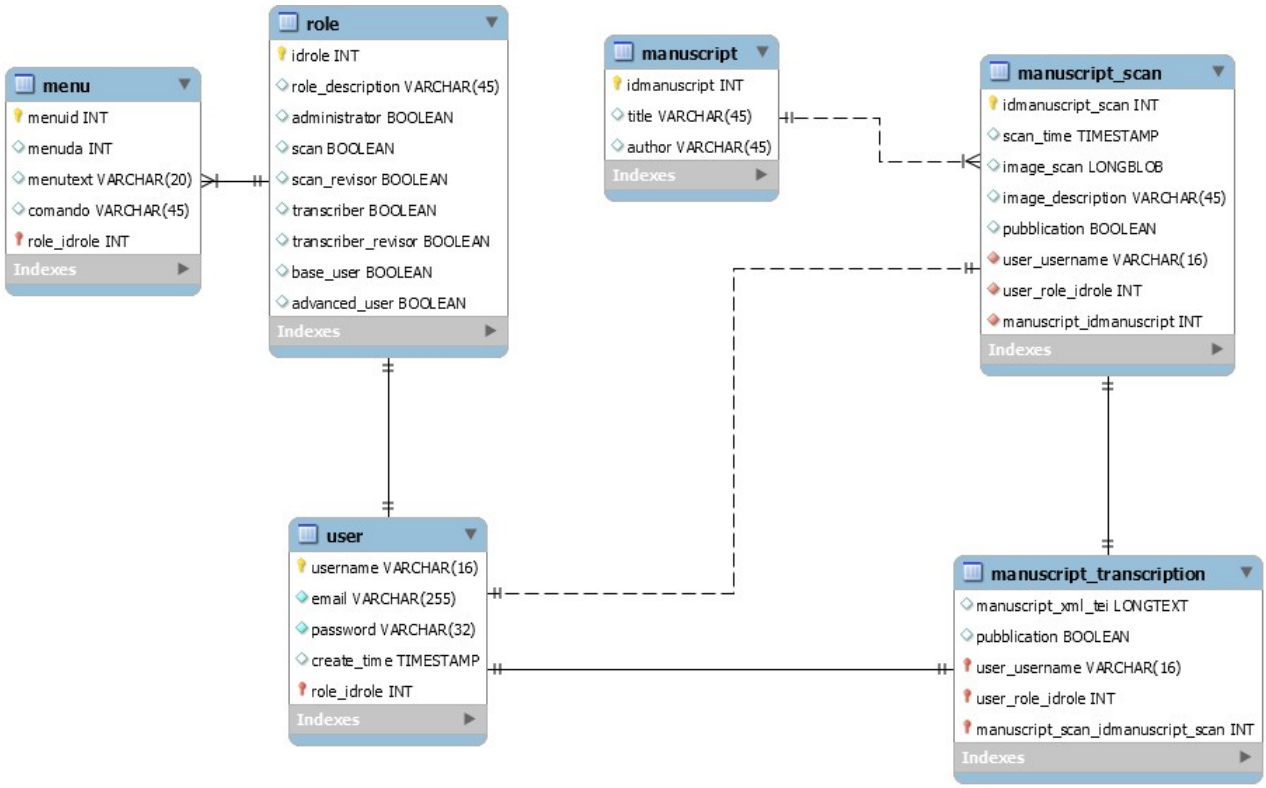


Figura 2 Schema ER del database MySQL utilizzato

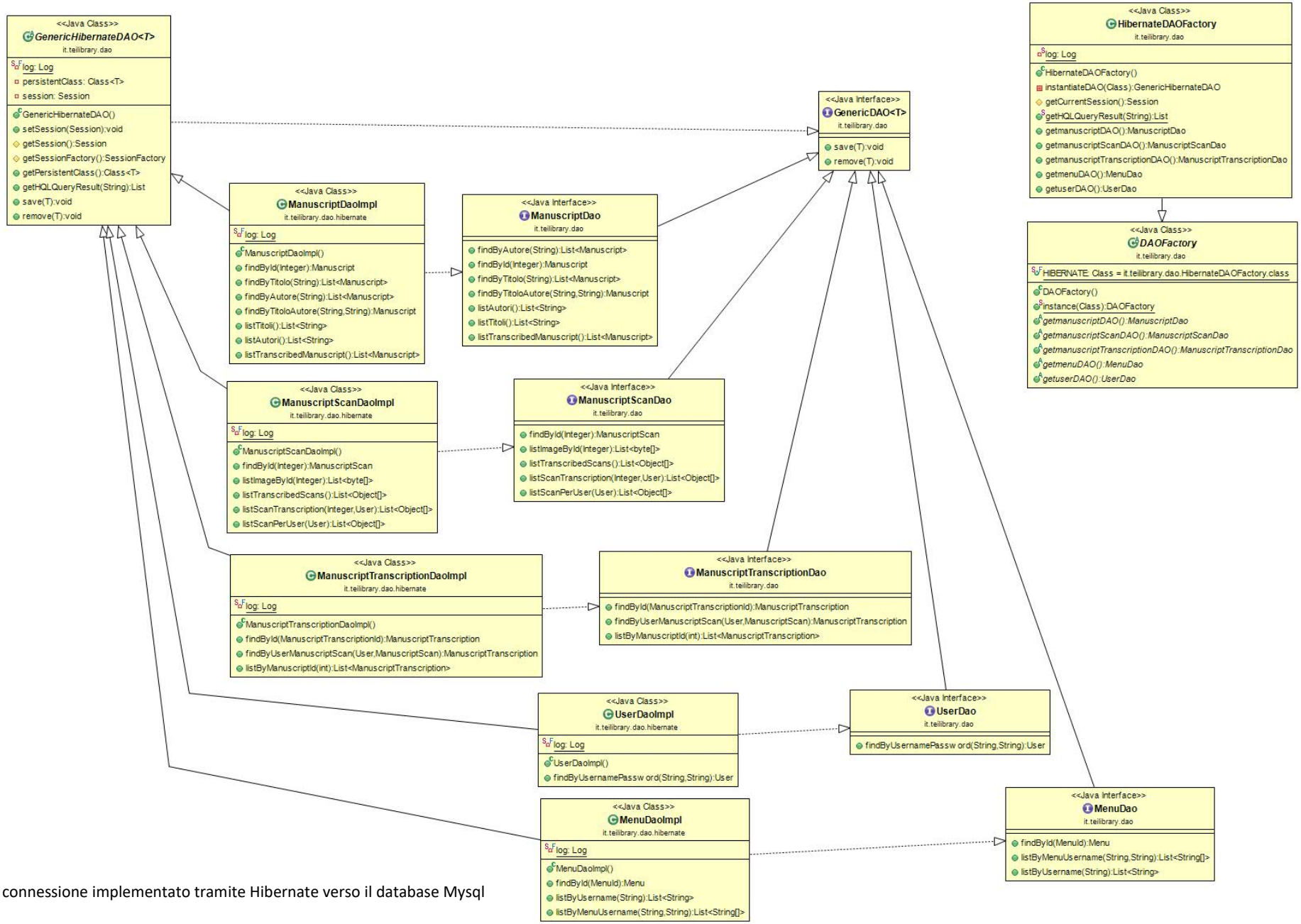


Figura 3 Dao connessione implementato tramite Hibernate verso il database Mysql

2.2 Descrizione dell'architettura

Use case templates

	Login
Descrizione	Richiede username e password per accedere all'applicazione
Importanza	Media
Evento scatenante	Vengono inserite le credenziali di accesso al sistema e si verifica l'accesso cliccando sul pulsante login
Uses	Genera il menu dell'utente riconosciuto, salvataggio sessione utente su variabile

	Visualizzazione elenco opere
Descrizione	Visualizza un elenco di tutte le opere trascritte a sistema
Importanza	Media
Evento scatenante	Viene richiamata la voce di menu che consente la visualizzazione delle opere trascritte
Uses	Verifica funzionalità in base al profilo utente

	Esportazione documento formato TEI
Descrizione	Scegliendo un'opera dalla visualizzazione elenco opere, è possibile esportare l'opera selezionata su un file di testo in formato TEI
Importanza	Media
Evento scatenante	Viene selezionata un'opera trascritta da esportare
Uses	Verifica funzionalità in base al profilo utente

	Lettura documento trascritto
Descrizione	Scegliendo un'opera dalla visualizzazione elenco opere, è possibile visionarla sulla maschera
Importanza	Media
Evento scatenante	Viene selezionata un'opera trascritta da esportare
Uses	Verifica funzionalità in base al profilo utente

	Acquisizione documento
Descrizione	Consente di caricare un documento in formato immagine, dopo aver selezionato l'opera e l'autore che si vuole caricare a sistema
Importanza	Media
Evento scatenante	Viene caricata a sistema un'immagine acquisita che compone l'opera da trascrivere
Uses	Verifica funzionalità in base al profilo utente

	Revisione acquisizione documento
Descrizione	Un documento da trascrivere può essere revisionato da un utente con i privilegi adatti, se non ritenuto eventualmente corretto per la trascrizione, cancellandolo
Importanza	Media
Evento scatenante	Viene selezionata un'acquisizione di una pagina del documento
Uses	Verifica funzionalità in base al profilo utente

	Pubblicazione per trascrizione
Descrizione	Un documento acquisito e revisionato, viene pubblicato per poter essere trascritto
Importanza	Media
Evento scatenante	Viene selezionata un'acquisizione di una pagina del documento
Uses	Verifica funzionalità in base al profilo utente

	Trascrizione documento
Descrizione	Un documento acquisito e pubblicato, può essere trascritto utilizzando l'apposito editor di testo
Importanza	Media
Evento scatenante	Viene selezionata un'acquisizione di una pagina del documento da trascrivere
Uses	Verifica funzionalità in base al profilo utente

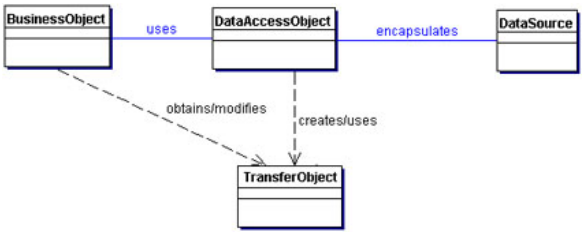
	Revisione trascrizione
Descrizione	Un documento trascritto viene revisionato, da un utente con i diritti necessari, per poter essere pubblicato come parte integrante dell'opera
Importanza	Media
Evento scatenante	Viene selezionata un'acquisizione di una pagina del documento trascritto
Uses	Verifica funzionalità in base al profilo utente

	Pubblicazione per consultazione
Descrizione	Un documento trascritto e revisionato, viene pubblicato per la consultazione tramite la funzionalità di lettura
Importanza	Media
Evento scatenante	Viene selezionata una trascrizione di una pagina del documento trascritto
Uses	Verifica funzionalità in base al profilo utente

Si utilizzerà un database MySQL per poter memorizzare tutti i dati, sia immagini acquisite che le trascrizioni, di cui abbiamo riportato il diagramma ER in figura 2.

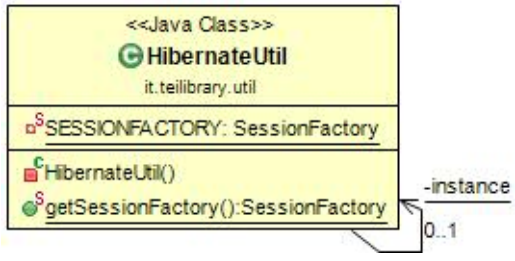
2.3 Descrizione delle scelte e strategie adottate

In figura 3 viene mostrato il class diagram utilizzato per modellare il database tramite Hibernate. E' stato utilizzato un **Data Access Object (DAO)** per astrarre e incapsulare tutti gli accessi al data source. Il DAO gestisce la connessione al data source per ottenere e storicizzare i dati.



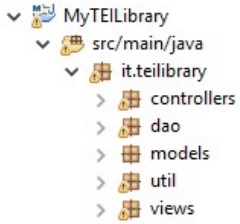
Il DAO implementa il meccanismo di accesso richiesto per lavorare con la sorgente dati. L'origine dati potrebbe essere un archivio permanente come un RDBMS, come nel nostro caso, file xml, o un semplice file di testo. Il componente business che si basa sul DAO, utilizza l'interfaccia più semplice esposta dal DAO per il client. Il DAO nasconde completamente la fonte dei dati e i dettagli di implementazione. Poiché l'interfaccia esposta dal DAO per il client non cambia quando i sottostanti cambiamenti di implementazione delle origini dati, questo modello permette al DAO per adattarsi a diversi sistemi di archiviazione. Essenzialmente, il DAO funge da adapter tra il componente e la sorgente di dati.

Il pattern DAO è stato reso quindi molto flessibile adottando un pattern di tipo **Abstract Factory**.
Il class diagram in fig. 3 del progetto implementato, mostra un factory DAO di base (factoryDAO), che è una classe astratta la quale viene ereditata ed implementata da una classe concreta DAO (HybernateDAOFactory) per supportare l'accesso e l'archiviazione specifica per l'implementazione, ma potrebbe comunque essere implementata un qualsiasi altro tipo di connessione sullo stesso database o su altri supporti, come precedentemente descritto.



Abbiamo usato il **Singleton Factory pattern** nella classe di utilità per la creazione del session factory, per la connessione al database tramite jdbc per Hibernate, in modo da non stabilire più di una connessione per ogni istanza del progetto.

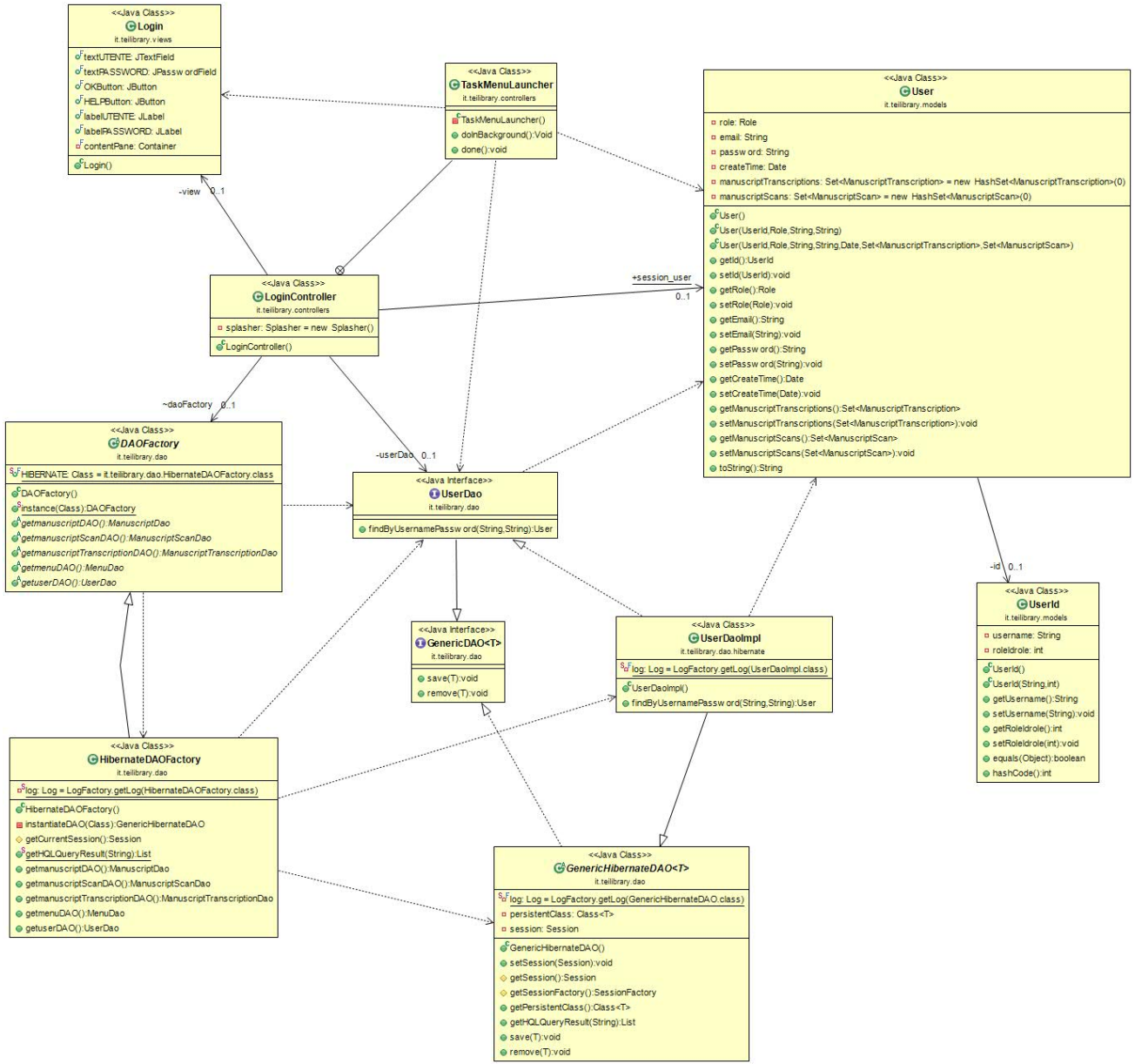
Adottiamo un approccio **architetturale di tipo MVC**, modello vista controllo, separando quindi la logica di presentazione dalla logica di business. Abbiamo quindi diviso tramite package gli strati dell'architettura.



- **Modello**: Il modello gestisce il comportamento ed i dati del dominio applicativo, risponde alle richieste di informazioni sul suo stato (le viste), e risponde alle istruzioni per modificare lo stato (i controller).
- **Vista**: La vista gestisce la visualizzazione delle informazioni.
- **Controller**: Il controller interpreta gli input di mouse e tastiera da parte dell'utente, informando il modello e / o la vista a seconda dei casi.

È importante notare che sia la vista e il controller dipendono dal modello. Tuttavia, **il modello non dipende né la vista né dal controller**. Questo è uno dei principali vantaggi della separazione. Questa separazione permette di costruire e testare il modello indipendente dalla rappresentazione visiva.

3.1 Modelli rappresentanti l'object design con classi, interfacce, membri



Nel class diagram mostriamo la divisione tra i componenti del **pattern architetturale MVC**, distinguendo la logica di presentazione dalla logica di business.

Inoltre distinguiamo nello specifico, l'utilizzo di un **abstract factory pattern** nell'utilizzo di un DAOFactory, che il controller utilizza per istanziare la connessione al database ed eseguire le operazioni CRUD con la base di dati sottostante, di qualunque natura esso sia (nel nostro progetto abbiamo utilizzato Hibernate tramite connessione JDBC su database MySQL).

Notiamo come il controller (in questa immagine il LoginController) non conosce i dettagli implementativi delle classi concrete che si occupano di interrogare il database.

Figura 4 Class diagram login

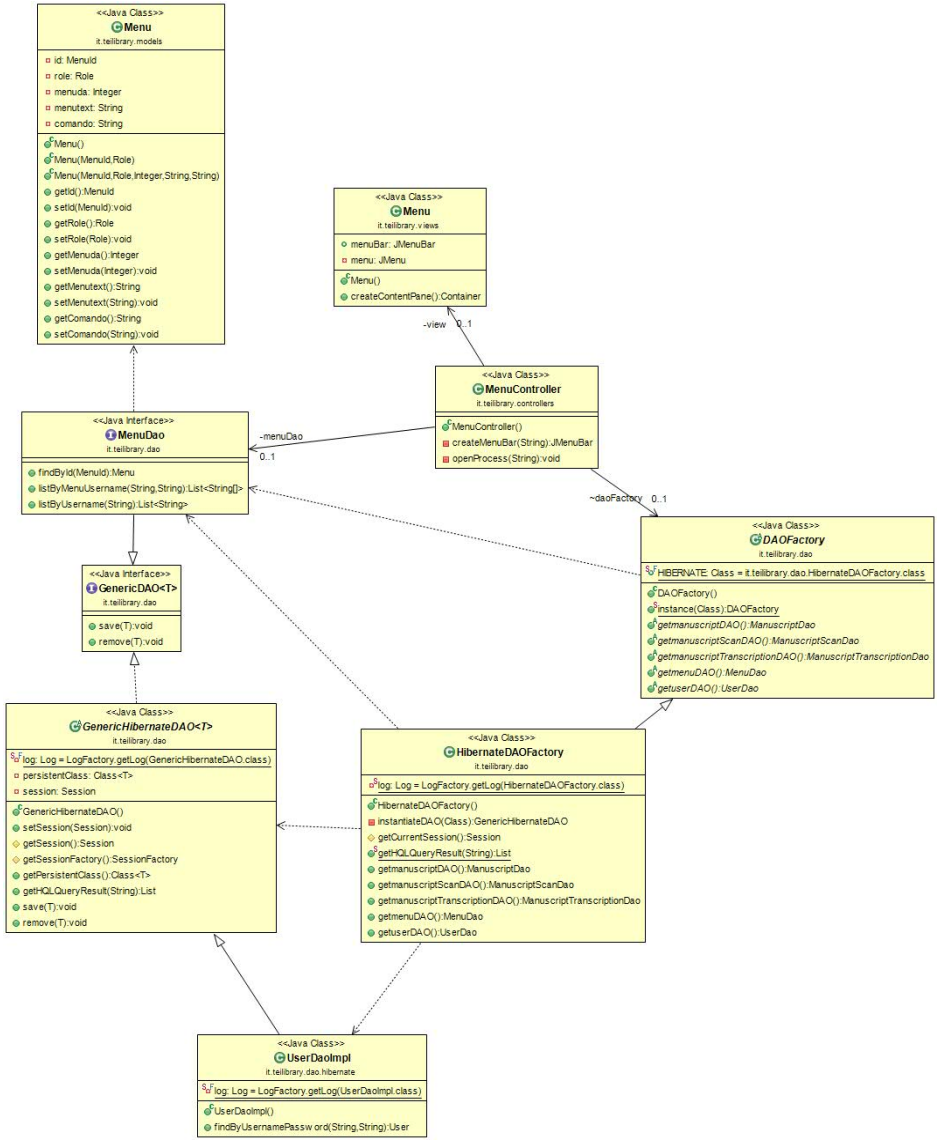


Figura 5 Class diagram costruzione menu e per il lancio applicazioni utente

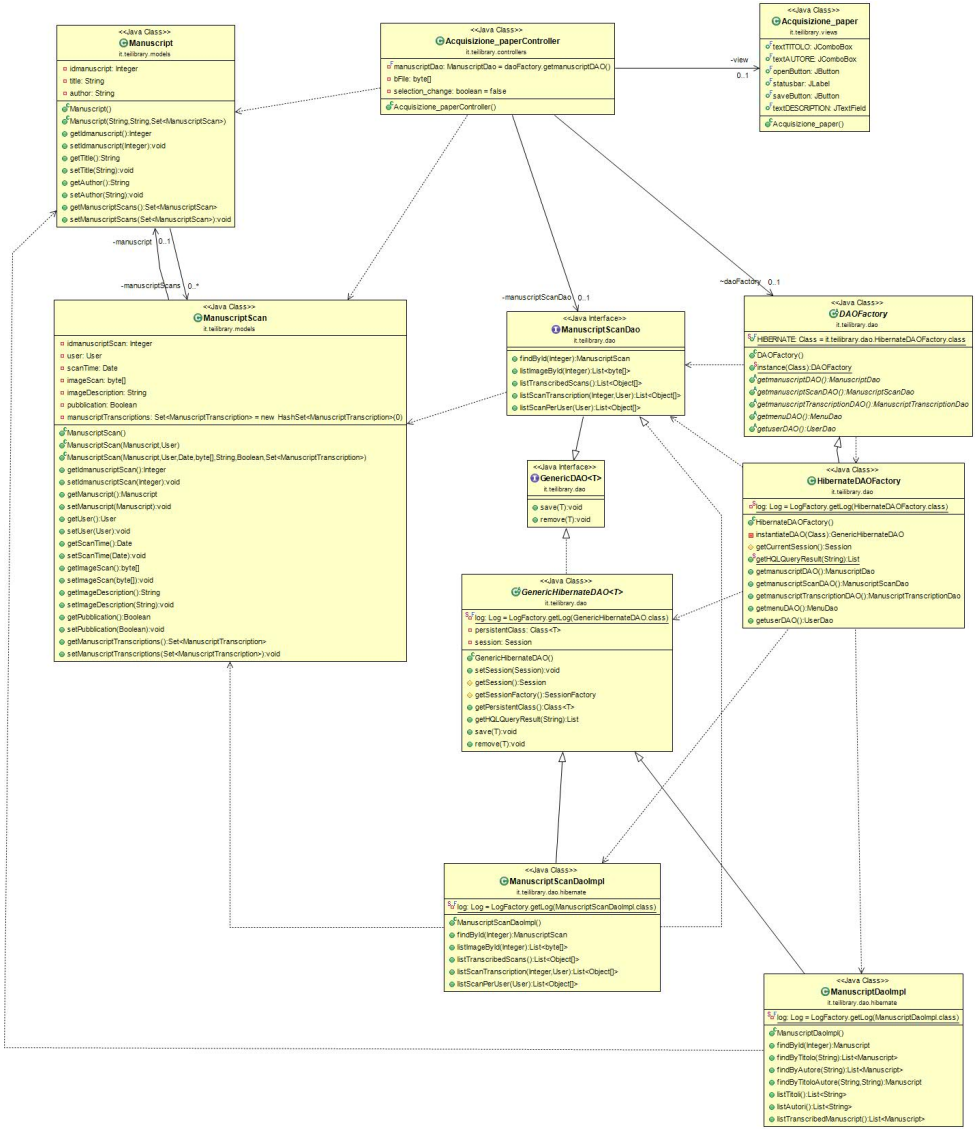


Figura 6 Class diagram maschera acquisizione immagini da scansione e attribuzione informazioni accessorie del manoscritto

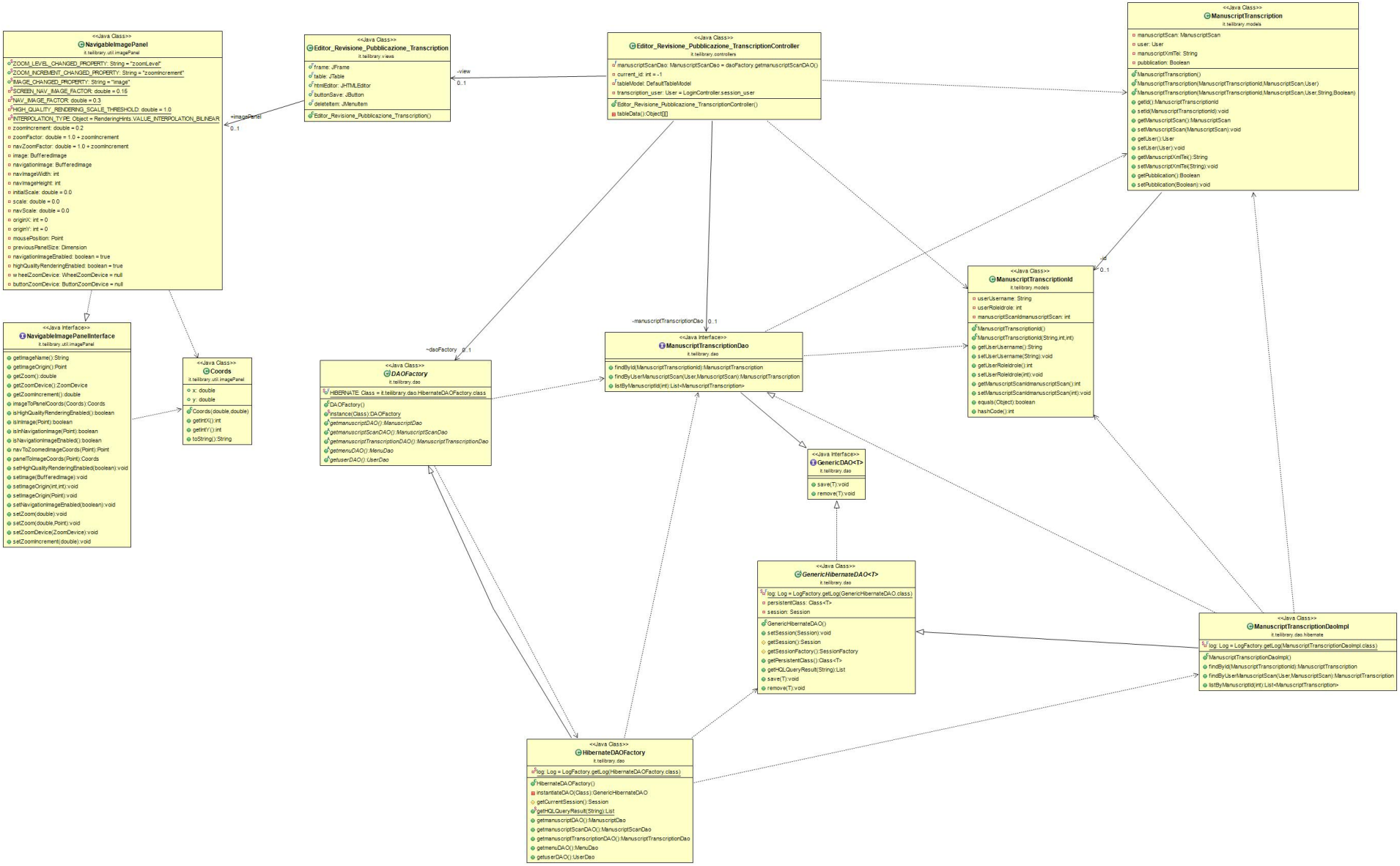


Figura 7 Class Diagram per la visualizzazione immagine acquisita e editor per trascrizione testo

3.2 Descrizione dei dettagli di design scelti

Si è scelto di creare, dato l'utilizzo di diversi utenti, una maschera di login per poter identificare i vari ruoli. Di conseguenza è stato modellato sul DB una tabella role, con una serie di campi booleani per poter mappare i vari ruoli utente. Inoltre abbiamo la necessità di utilizzare dei menu specifici per il ruolo utente, dove abiliteremo solamente le voci di diretto interesse per l'utente, evitando le maschere dove non si hanno diritti.

Utilizziamo quindi una serie di interfacce grafiche da lanciare quando si sceglierà la voce di menu specifica.

Avremo bisogno in particolare di una serie di funzioni di utilità, riutilizzabili tra le interfacce grafiche, tra cui un **pannello immagine navigabile** e con la possibilità di poter zoommare i dettagli dell'immagine acquisita e di un **editor di testo** per poter trascrivere in maniera quanto più simile all'immagine originale, per poter in seguito essere facilmente visualizzato al momento della lettura del documento trascritto e quindi **esportabile in formato TEI**. Abbiamo quindi bisogno di una classe di manipolazione del contenuto trascritto in formato TEI.

La struttura dei class diagram mostrate da **figura 4 a figura 7** è pressoché la stessa per tutti i class rappresentati.

Questo perché il **pattern DAO** si occupa di generalizzare il tipo di accesso e interrogazione verso il database, mentre il **pattern MVC** distingue i vari modelli comportamentali dell'applicazione. Ne risulta quindi che lo strato di **business**, di **presentazione** e **persistenza** siano ben identificabili e standardizzati come si evince osservandone la struttura stessa.

Ogni class diagram dell'applicazione qui descritta presenta quindi un'implementazione per lo strato di presentazione, una per il controller e una o più, a seconda delle informazioni da trattare, per il layer di persistenza.

Utilizziamo **Maven** per poter facilmente mappare le dipendenze e poter facilmente importare il progetto.

Utilizziamo i **thread** per poter lanciare il task di costruzione delle voci di menu utente.

Utilizziamo una tecnologia **Hibernate** per poter modellare il database MySQL riportato sullo schema ER precedente.
Questo ci consente di mostrare l'utilizzo dei **javaBean**, caratteristiche proprie di Hibernate, ma anche la manipolazione dei tipi di dati applicato sull'estrazione come risultato delle interrogazioni su DB generalmente di tipo **generic List**.

Abbiamo dichiarato quindi una classe di utilità HibernateUtil, per poter effettuare la connessione jdbc sul DB, dichiarata sul file di configurazione hibernate.cfg.xml, di tipo **Singleton** per impedire che possano essere instaurate più di una connessione di sessione SessionFactory.

Ci siamo serviti di una classe di utilità per la manipolazione delle immagini caricate (NavigableImagePanel) a sistema estendendo la classe JPanel , e implementando la relativa interfaccia dichiarata.

Infine abbiamo utilizzato una classe statica per poter esportare su un file di testo le elaborazioni in formato TEI.

Di seguito una serie di interfacce grafiche realizzate per il sistema.

Figura 8 Machera di login utente

La maschera consente l'identificazione dell'utente, restituendo un errore se il login non va a buon fine.

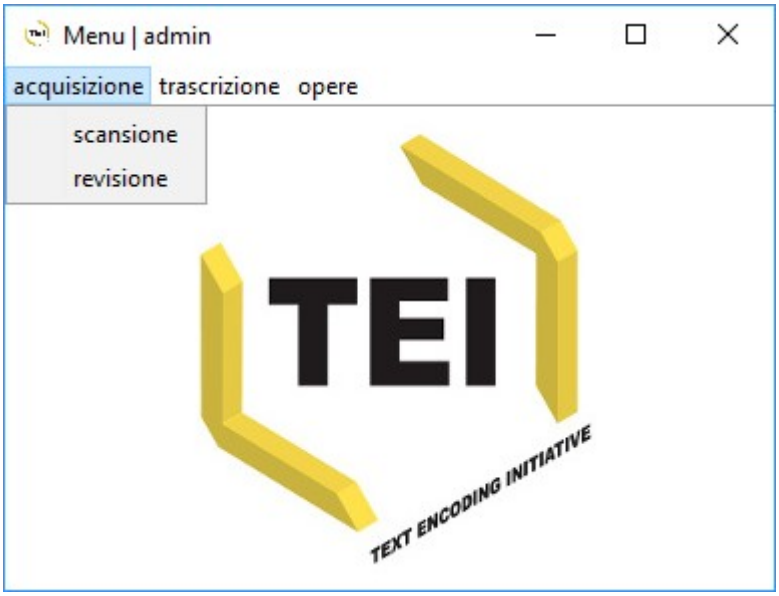


Figura 9 Menu personalizzato per l'utente connesso

Il menu viene costruito in base all'utente riconosciuto, mostrando tutte le funzionalità collegate al suo ruolo profilo.

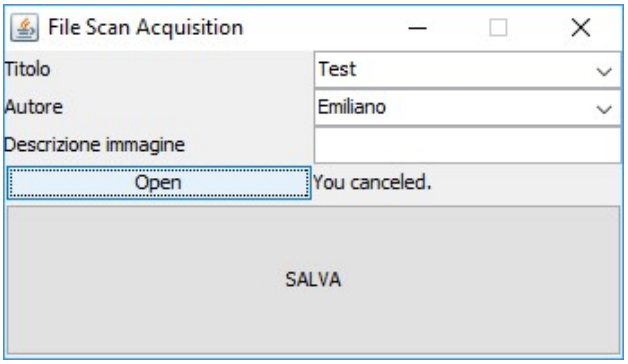


Figura 10 Maschera di caricamento immagini acquisite

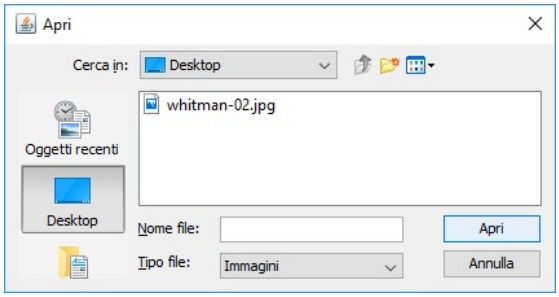


Figura 11 JFileChooser filtrato per selezionare le immagini da caricare

La funzionalità consente di scegliere un titolo e un autore, già a sistema o di inserirne di nuovi, una breve descrizione del contenuto che si sta caricando e un pulsante di selezione file filtrato. Salvando i dati, verrà salvato tutto il contenuto qui inserito, immagine inclusa, sul database.

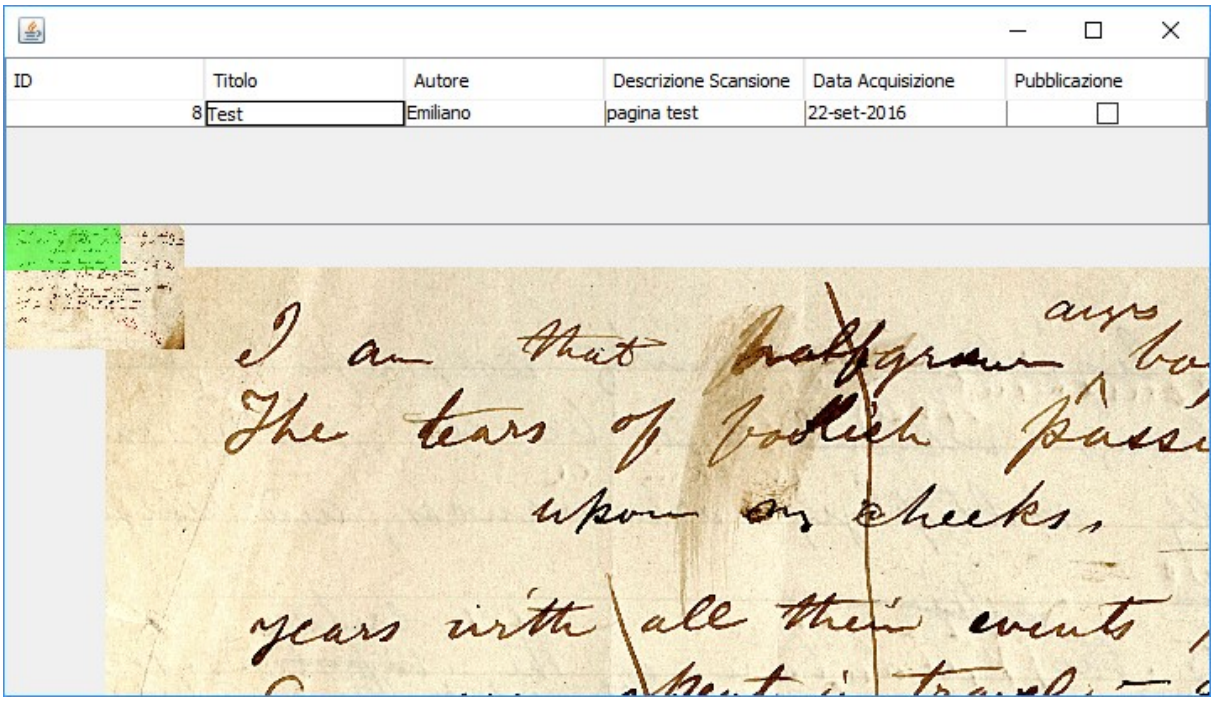


Figura 12 Maschera di revisione immagine con pannello immagine navigabile

Una volta caricate una o più immagini che compongono l'opera, è possibile revisionare, navigando e ingrandendo l'immagine, e se non conforme cancellarla, o in caso contrario pubblicarla per la trascrizione.

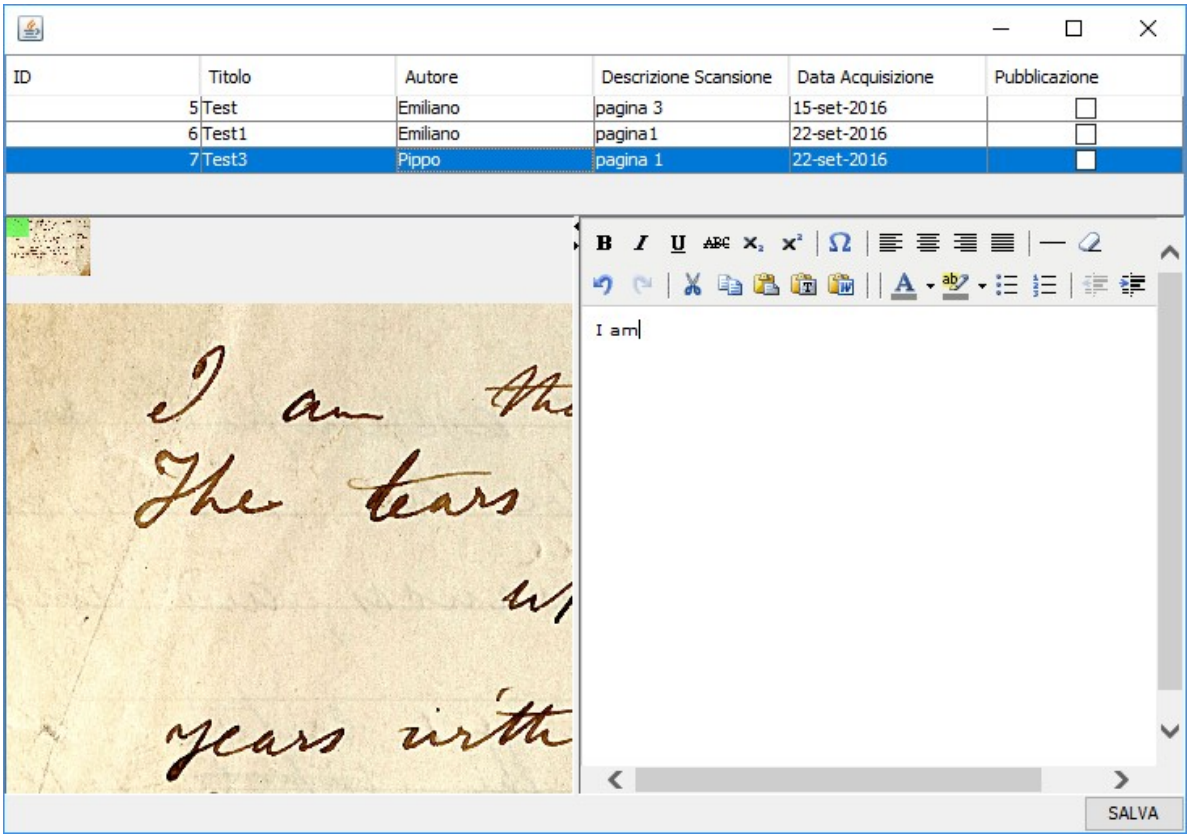


Figura 13 Maschera di trascrizione, revisione e pubblicazione in base ai diritti utente. In dettaglio l'editor di testo con le varie funzionalità di formattazione

Selezionando una scansione acquisita, caricheremo la relativa immagine, sempre navigabile e un editor di testo che consente la trascrizione. Una volta terminato possiamo salvare il contenuto trascritto e , se l'utente possiede i diritti di revisione, revisionare, salvare e pubblicare la trascrizione.

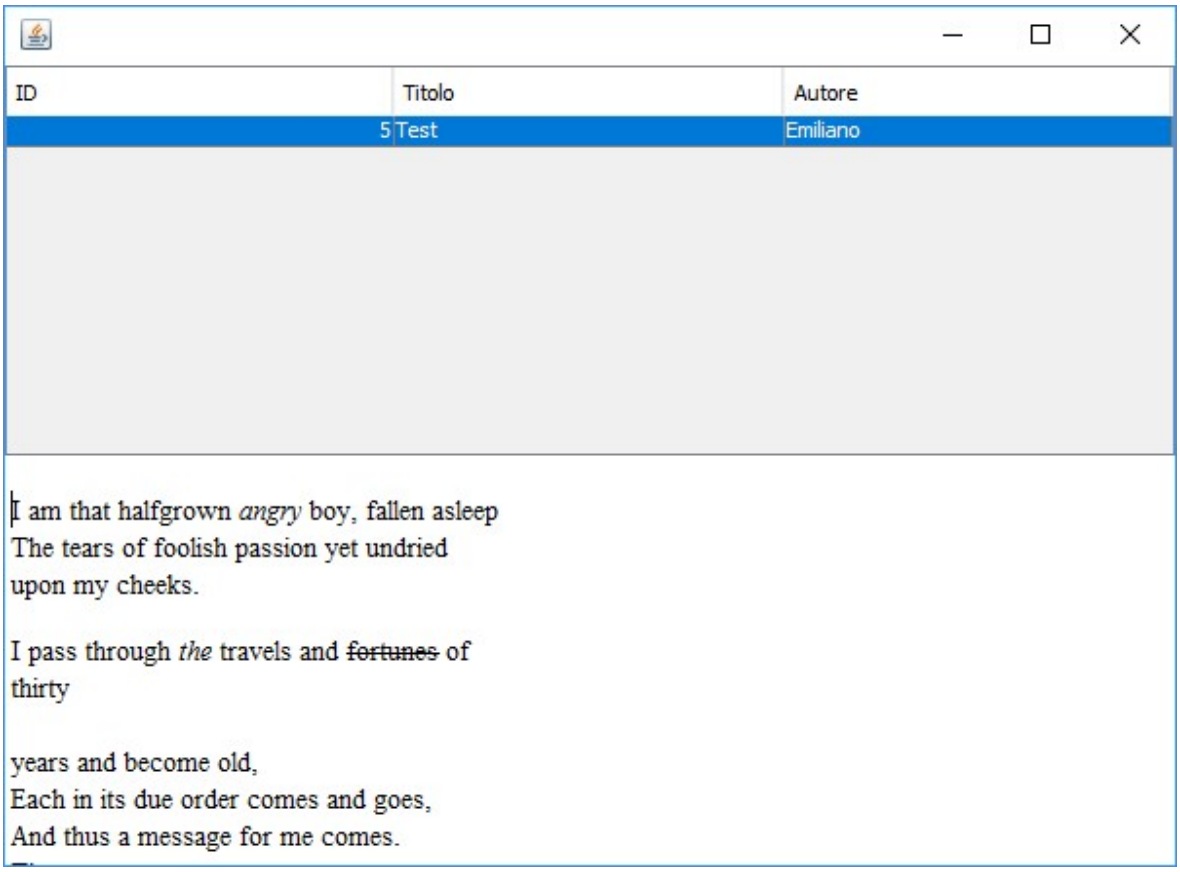


Figura 14 Maschera di visualizzazione elenco opere pubblicate e lettura testo trascritto. E' possibile esportare su un file di testo formato TEI valido, la trascrizione selezionata.

La maschera consente la visualizzazione per l-utente base di visualizzare l-elenco delle opere trascritte, mentre l'utente con diritti avanzati, può sia leggerne il contenuto trascritto sia esportare il contenuto in formato TEI valido su un file di testo.