Progetto Object-Oriented Software Design

Progetto “Biblioteca Digitale”

A.A. 2017/2018

Cicerone Loreto – De Cesaris Marco – Caruso Simone

1)Specifiche:

Il progetto è un’estensione della traccia fornita nell’a.a. 2015/16 e si propone di realizzare una biblioteca digitale di testi e studi che contribuiscono alla formazione della cultura all’interno dell’Università degli Studi dell’Aquila. Una biblioteca digitale è uno spazio in cui mettere insieme collezioni, servizi e persone a supporto dell'intero ciclo di vita di creazione, uso, preservazione di dati, informazione e conoscenza. Lo scopo di questo progetto è di consentire la consultazione dei manoscritti che devono essere digitalizzati e che costituiscono un patrimonio bibliografico antico per un totale di 60.000 carte (ms.sec.XV-XIX) contenenti memorie storiche della città dell’Aquila. Il sistema si divide nei seguenti sottosistemi:

1. Viewer
2. Uploader
3. Transcriber
4. Manager
5. Administrator

I sottosistemi sono indipendenti tra loro ma comunicano al fine di realizzare i loro scopi.

In particolare, vengono descritti come segue.

Viewer:

Tale parte del sistema consente la consultazione delle opere digitali a utenti registrati. Consente la ricerca nel catalogo per metadati (descritti in seguito) oppure all’interno del testo della trascrizione. Le opere possono essere suddivisi in categorie. Appena si sceglie un’opera, verrà visualizzata con una schermata che avrà sulla destra il testo della trascrizione (se disponibile) e sulla sinistra l’immagine della pagina dell’opera che si sta visualizzando, sarà possibile sfogliare le pagine tramite un paginatore. Alcuni utenti con particolari privilegipossono effettuare il download dell’opera. L’utente può fare richiesta tramite un modulo per essere collaboratore del sistema (trascrittore). Gli utenti possono accedere al loro profilo personale dove saranno visualizzati i dati inseriti nella registrazione, tra cui: titolo di studio, professione, indirizzo, email, etc.

Uploader:

Ogni opera è formata da più immagini (scansioni), ognuna delle quali rappresenta una pagina del manoscritto. Per ogni opera vengono caricati dei metadati (titolo, anno, …). Al fine di rendere agevole il caricamento delle immagini e il successivo controllo, per ogni opera si vuole fornire la visualizzare sia tutte le pagine in miniatura e di una pagina per volta da scorrere con un paginatore. La digitalizzazione viene controllata da supervisori all’acquisizione per assicurarne la correttezza (ad esempio, in accordo con standard richiesti) e la qualità.

Transcriber:

Ogni opera acquisita deve essere trasformata in un testo digitale, ciò avviene attraverso operazioni di trascrizioni in formato TEI (Text Encoding Initiative). Le trascrizioni sono digitate manualmente attraverso un text editor TEI integrato. Quindi ogni immagine (pagina) avrà il corrispondente testo associato. Più trascrittori possono lavorare sulla stessa pagina, è necessario sincronizzare le modifiche. Le trascrizioni sono oggetto di revisione da parte di revisori alle trascrizioni.

Manager:

Questo sottosistema gestisce le assegnazioni, ovvero consente di assegnare parte di un’opera (1 o più immagini) a 1 o più trascrittori. Inoltre, consente di revisionare le trascrizioni concluse, di effettuare correzioni e validazione. E’ possibile anche riassegnare delle pagine ai trascrittori. Consente la pubblicazione delle trascrizioni e delle opere (solo immagini). Gestisce i livelli dei trascrittori (ogni trascrittore ha un livello 1-5 in base alla sua esperienza). Consente la supervisione dell’acquisizione immagini.

Administrator:

Gestione in back-end di tutto il sistema: anagrafica utenti, opere, etc.

1.1) Analisi dei requisiti:

Individuazione dei concetti fondamentali:

Il progetto è un’estensione della traccia fornita nell’a.a. 2015/16 e si propone di realizzare una biblioteca digitale di testi e studi che contribuiscono alla formazione della cultura all’interno dell’Università degli Studi dell’Aquila. Una biblioteca digitale è uno spazio in cui mettere insieme collezioni, servizi e persone a supporto dell'intero ciclo di vita di creazione, uso, preservazione di dati, informazione e conoscenza. Lo scopo di questo progetto è di consentire la consultazione dei manoscritti che devono essere digitalizzati e che costituiscono un patrimonio bibliografico antico per un totale di 60.000 carte (ms.sec.XV-XIX) contenenti memorie storiche della città dell’Aquila. Il sistema si divide nei seguenti sottosistemi:

1. Viewer
2. Uploader
3. Transcriber
4. Manager
5. Administrator

I sottosistemi sono indipendenti tra loro ma comunicano al fine di realizzare i loro scopi.

In particolare, vengono descritti come segue.

Viewer:

Tale parte del sistema consente la consultazione delle opere digitali a utenti registrati. Consente la ricerca nel catalogo per metadati (descritti in seguito) oppure all’interno del testo della trascrizione. Le opere possono essere suddivisi in categorie. Appena si sceglie un’opera, verrà visualizzata con una schermata che avrà sulla destra il testo della trascrizione (se disponibile) e sulla sinistra l’immagine della pagina dell’opera

che si sta visualizzando, sarà possibile sfogliare le pagine tramite un paginatore. Alcuni utenti con particolari privilegi possono effettuare il download dell’opera. L’utente può fare richiesta tramite un modulo per essere collaboratore del sistema (trascrittore). Gli utenti possono accedere al loro profilo personale dove saranno visualizzati i dati inseriti nella registrazione, tra cui: titolo di studio, professione, indirizzo, email, etc.

Uploader:

Ogni opera è formata da più immagini (scansioni), ognuna delle quali rappresenta una pagina del manoscritto. Per ogni opera vengono caricati dei metadati (titolo, anno, …). Al fine di rendere agevole il caricamento delle immagini e il successivo controllo, per ogni opera si vuole fornire la visualizzazione sia tutte le pagine in miniatura e di una pagina per volta da scorrere con un paginatore. La digitalizzazione viene controllata da supervisori all’acquisizione per assicurarne la correttezza (ad esempio, in accordo con standard richiesti) e la qualità.

Transcriber:

Ogni opera acquisita deve essere trasformata in un testo digitale, ciò avviene attraverso operazioni di trascrizioni in formato TEI (Text Encoding Initiative). Le trascrizioni sono digitate manualmente attraverso un text editor TEI integrato. Quindi ogni immagine (pagina) avrà il corrispondente testo associato. Più trascrittori possono lavorare sulla stessa pagina, è necessario sincronizzare le modifiche. Le trascrizioni sono oggetto di revisione da parte di revisori alle trascrizioni.

Manager:

Questo sottosistema gestisce le assegnazioni, ovvero consente di assegnare parte di un’opera (1 o più immagini) a 1 o più trascrittori. Inoltre, consente di revisionare le trascrizioni concluse, di effettuare correzioni e validazione. E’ possibile anche riassegnare delle pagine ai trascrittori. Consente la pubblicazione delle trascrizioni e delle opere (solo immagini). Gestisce i livelli dei trascrittori (ogni trascrittore ha un livello 1-5 in base alla sua esperienza). Consente la supervisione dell’acquisizione immagini.

Administrator:

Gestione in back-end di tutto il sistema: anagrafica utenti, opere, etc.

Glossario dei termini:

Termine	Descrizione
Viewer	Utente correttamente registrato nel sistema e con la quale può liberamente interagire entro i suoi limiti.
Opere	Manoscritti che possono essere visionati dagli utenti registrati nel sistema.
Ricerca	Il sistema deve offrire l’opportunità all’utente di effettuare ricerche di Manoscritti per metadati.
Trascrizione	Trascrizione di un’immagine eseguita attraverso un TEI in

	caratteri digitali.
Immagine	Immagine di una certa pagina del Manoscritto.
Download	Un utente con particolari privilegi può effettuare lo scaricamento di una certa immagine di una pagina del Manoscritto.
Profilo Personale	L’utente deve avere la possibilità di poter visionare i propri dati, opportunatamente registrati all’interno del suo profilo personale, accessibile dalla piattaforma.
Uploader	Utente che ha il compito di caricare immagini di pagine di un certo Manoscritto.
Metadati	Campi che identificano un certo Manoscritto all’interno del sistema.
Transcriber	Utente che ha la possibilità di poter trascrivere in caratteri digitali una certa immagine caricata all’interno della piattaforma utilizzano un TEI.
T.E.I.	Text-Editor opportunatamente creato con il quale è possibile trascrivere le immagini caricate.
Manager	Uno dei più importanti sottosistemi della piattaforma. Egli ha il dovere di revisionare il lavoro svolto dall’uploader e dal transcriber.
Assegnazioni	Il manager, tra i suoi compiti, deve assegnare delle immagini ai transcriber, i quali ne effettueranno la trascrizione.
Revisione	Attività svolta dal manager, la quale consiste nell’accettare o rifiutare immagini o trascrizioni inserite.
Ri/assegnare	Attività svolta dal manager, la quale consiste nell’assegnare o nel riassegnare immagini

	caricate nel sistema a più transcriber, che quindi possono trascrivere la stessa immagine. Il lavoro sarà valutato direttamente dal Manager.
Administrator	Amministratore del sistema, il quale si occupa della gestione dell’intera piattaforma.

1.2) Documento dei requisiti:

Il sistema si pone come obiettivo quello di offrire un servizio che consenta la consultazione, da parte degli utenti, di antichi manoscritti, il tutto come una biblioteca virtuale. L’utente deve necessariamente effettuare una registrazione nella biblioteca virtuale, mettendo tutti i vari campi consoni ad una corretta registrazione, ed in seguito può effettuare un login. Tra gli utenti che si iscrivono possono esserci utenti con particolari privilegi, che, in particolare, hanno la possibilità di effettuare il download di un’immagine del manoscritto che desiderano. I dati degli utenti possono essere consultati dagli stessi quando si vuole, grazie ad una sezione “profilo utente” fornita direttamente dal sistema. L’utente, inoltre, riempiendo un apposito modulo, può avere la possibilità di diventare un transcriber. Il sistema prevede la visualizzazione del manoscritto nel seguente modo: Una volta scelto il manoscritto da voler visualizzare saranno presenti a sinistra e destra, rispettivamente, l’immagine di una pagina del manoscritto e la rispettiva trascrizione al suo fianco, quest’ultima effettuata dal transcriber. Le immagini sono caricate dall’uploader, il quale ha un ruolo diverso rispetto al transcriber, infatti esso è incaricato solo del caricamento delle immagini. Il sistema consente di visualizzare quest’ultime sia in miniatura, sia per intero, con un opportuno paginatore che consente di andare avanti e indietro nello scorrimento delle immagini delle pagine. L’uploader ha a disposizione 4 tipi di metadati da caricare per ogni opera: Autore, titolo, anno, genere. Una volta eseguito questo passo, le varie immagini saranno visionate dal manager, anch’esso un altro sottosistema, che ne esaminerà la correttezza. Una volta che le immagini caricate dall’uploader saranno accettare dal manager, sarà compito del transcriber elaborare le immagini traducendole in caratteri digitali.
Il sistema prevede che ogni opera avrà una o più pagine al suo interno, che possono essere visualizzate come mosaico oppure per intero. Tali pagine saranno così formate: A sinistra c’è l’immagine selezionata, mentre a destra c’è l’editor di testo dove è possibile trascrivere il testo dell’immagine in formato TEI. Ogni transcriber avrà un grado di esperienza ad esso associato, riportato in una scala da 1 a 5, e visualizzabile all’interno della sezione profilo fornita dal sistema. Le varie pagine vengono assegnate al transcriber grazie al manager, il quale può decidere anche di assegnare la stessa pagina a

transcriber diversi. Una volta effettuata la trascrizione della pagina, sarà compito del manager verificare la correttezza e la validazione della trascrizione. Spetterà quindi al manager verificare l’esattezza del lavoro svolto dall’uploader e dal transcriber. Il sistema prevede inoltre che la sezione utente, con tutti i vari campi che lo identificano, e la sezione delle opere, anch’esse verificate da campi come Autore, Titolo, Anno e genere, siano gestite da un amministratore della piattaforma. Il sottosistema più importante della piattaforma è però l’Amministratore, il quale ha il compito di gestire l’intera piattaforma.

I requisiti funzionali del progetto sono i seguenti, indicati da una scala di priorità da 1 a 5 (1: priorità minima, 5: priorità massima):

1) **Registrazione e Login**: L’utente deve registrarsi per poter usufruire dei vari servizi (Priorità 5).

2) **Univocità**: Nel sistema non possono comparire utenti con nickname uguali, evitando, quindi, ambiguità nel sistema stesso. (Priorità 5).

3) **Utente VIP**: Nel sistema possono esserci utenti con particolari privilegi che possono scaricare un’immagine di un’opera. (Priorità 1).

4) **Profilo Utente**: L’utente deve avere la possibilità di poter modificare i suoi campi e generalità (memorizzati in un opportuno Database). (Priorità 3).

5) **Diventa Transcriber**: Il sistema deve dare la possibilità agli utenti di poter diventare transcriber, quindi nel sistema ce ne deve essere almeno uno che si occupi della trascrizione. L’utente può diventare transcriber solo dopo aver compilato l’apposito modulo fornitogli, oppure promosso grazie al lavoro dell’amministratore (Priorità 5).

6) **Sezione visualizzazione opera**: Il sistema deve dare la possibilità di poter visualizzare le immagini dell’opera per intero, con relativa trascrizione affianco (Priorità 5).

7) **Declinazione immagine**: Il sistema deve fornire all’utente immagini di buona e non di scarsa qualità caricate dall’uploader, aspetto che può rendere difficile la trascrizione. Ne consegue che il manager può declinare il caricamento di immagini di bassa qualità (Priorità 4).

8) **Univocità dell’immagine**: Il sistema deve dare ad una immagine del manoscritto un numero che la identifichi univocamente all’interno dell’opera, compito svolto, in particolare, dall’uploader. (Priorità 4).

9) **Ricerca Opera**: Il sistema deve dare la possibilità all’utente di poter eseguire una ricerca dell’opera che a lui interessa (Priorità 5).

I requisiti non funzionali sono riportati di seguito:

1) **Usability**: Usabilità del sistema, ovvero la facilità per l’utente di imparare ad usare il sistema rapidamente, fornendo, quindi, un’interfaccia user-friendly.

2) **Reliability**: Affidabilità del sistema, ovvero la reazione a casi limite, garantendo le funzionalità messe a disposizione senza errori. Ad esempio il sistema deve poter bloccare la registrazione di un utente nel caso in cui i 2 abbiano nickname uguali.

3) **Performance**: Velocità di esecuzione del sistema, ovvero il lavoro che il sistema riesce ad eseguire in una data unità di tempo.

1.3) Attori del sistema:

Utente Loggato: L’utente è il primo attore che è messo in risalto dal sistema. Esso, come già specificato, per interagire con la piattaforma, deve necessariamente effettuare una registrazione, ed in seguito un login. Esso ha, inoltre, la possibilità di diventare un **Utente Vip,** e quest’ultimo, in particolare, ha la possibilità di poter scaricare le immagini dell’opera che più gli interessano. L’utente può consultare il suo profilo utente quando desidera, e può, in caso, modificare i suoi campi (generalità e altro) quando desidera. Inoltre, compilando un apposito modulo che apparirà sempre nella sua sezione **profilo utente**, ha la possibilità di diventare **Transcriber.** Il transcriber ha la possibilità di poter trascrivere le immagini di singole pagine dell’opera che gli vengono assegnate dal **manager** (descritto in seguito).

Transcriber: Il transcriber, come già descritto, è semplicemente un utente già iscritto nel sistema, il quale ha la possibilità di trascrivere opere. Il manager può assegnare l’immagine da caricare nel sistema a più transcriber, che quindi possono aver modificato più volte la trascrizione. Sarà compito sempre del **manager** scegliere quale opera sia più consona. Il transcriber inoltre ha associato un grado di esperienza che può variare da 1 a 5.

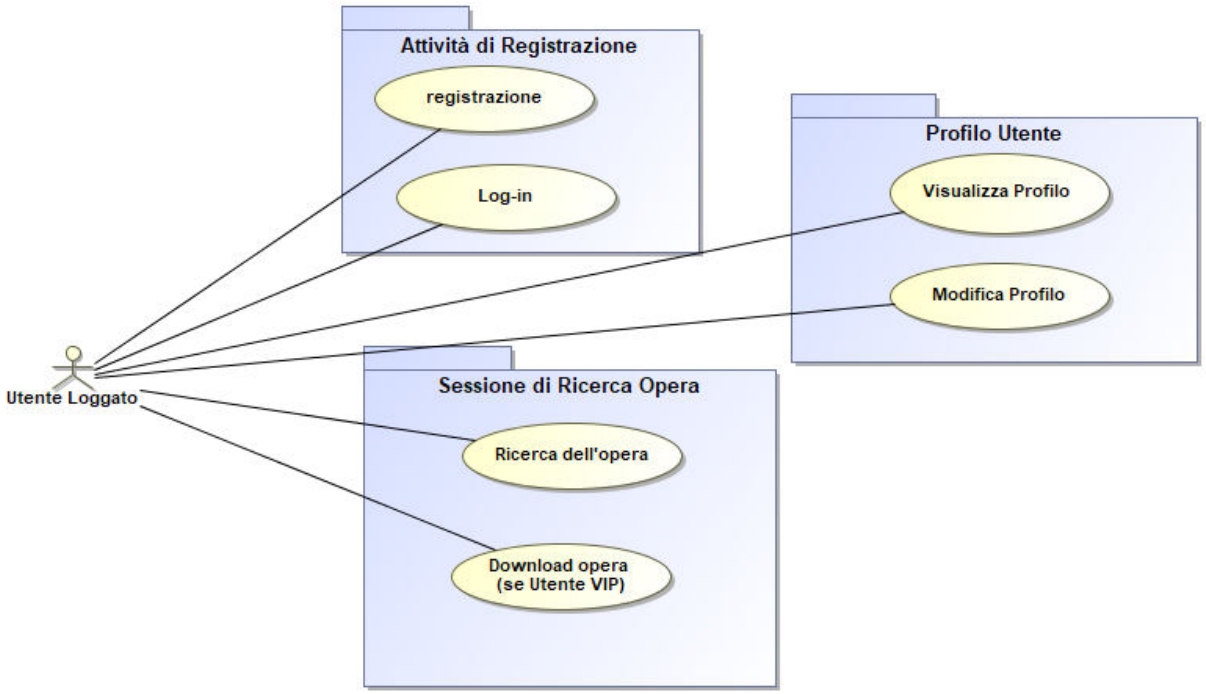
Manager: Il manager è uno degli attori più importanti del sistema. Cosi come un normale utente, egli ha la possibilità di modificare e visualizzare il suo profilo. Egli, inoltre, ha il dovere di dover gestire il grado di esperienza dei transcriber, di visionare la trascrizione delle opere effettuando nel caso anche correzioni, ma anche di accettare nel sistema immagini delle varie opere che si dovranno trascrivere (possibilmente immagini di almeno buona qualità). Questo aspetto del caricamento delle immagini, è svolto da un altro attore del sistema, **l’uploader.**

Uploader: Anche l’uploader può essere considerato come un utente della piattaforma. Il suo compito è quello di caricare immagini nel sistema. Ogni immagine caricata deve avere un numero che la identifichi univocamente all’interno del sistema. Inoltre, ogni immagine deve essere facilmente distinguibile all’interno dell’opera. A tal proposito si può pensare di identificare ogni immagine attraverso un numero che la identifichi univocamente all’interno dell’opera stessa.

Amministratore: L’amministratore è senza dubbio l’attore più importante all’interno del sistema, infatti ha la completa gestione dell’intera piattaforma. Egli ha la possibilità di eliminare utenti all’interno del sistema, interagendo con un database, e di promuovere gli stessi ad **Utenti Vip**, secondo opportuni privilegi che questi hanno. Ha anche la possibilità di modificare ed eliminare opere, anch’esse memorizzate all’intero del database.

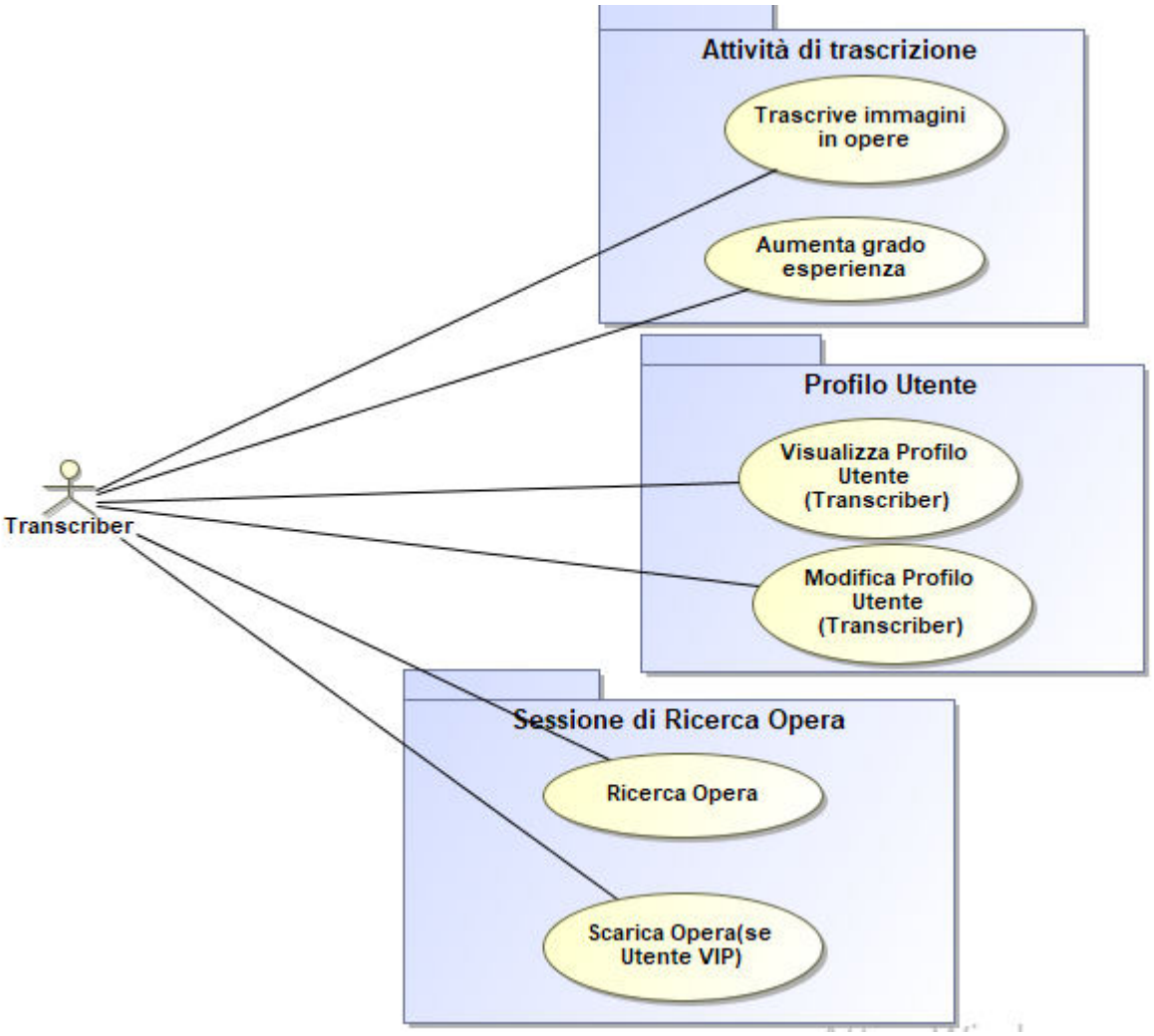
1.4) Use-Case:

Use-case per l’utente:



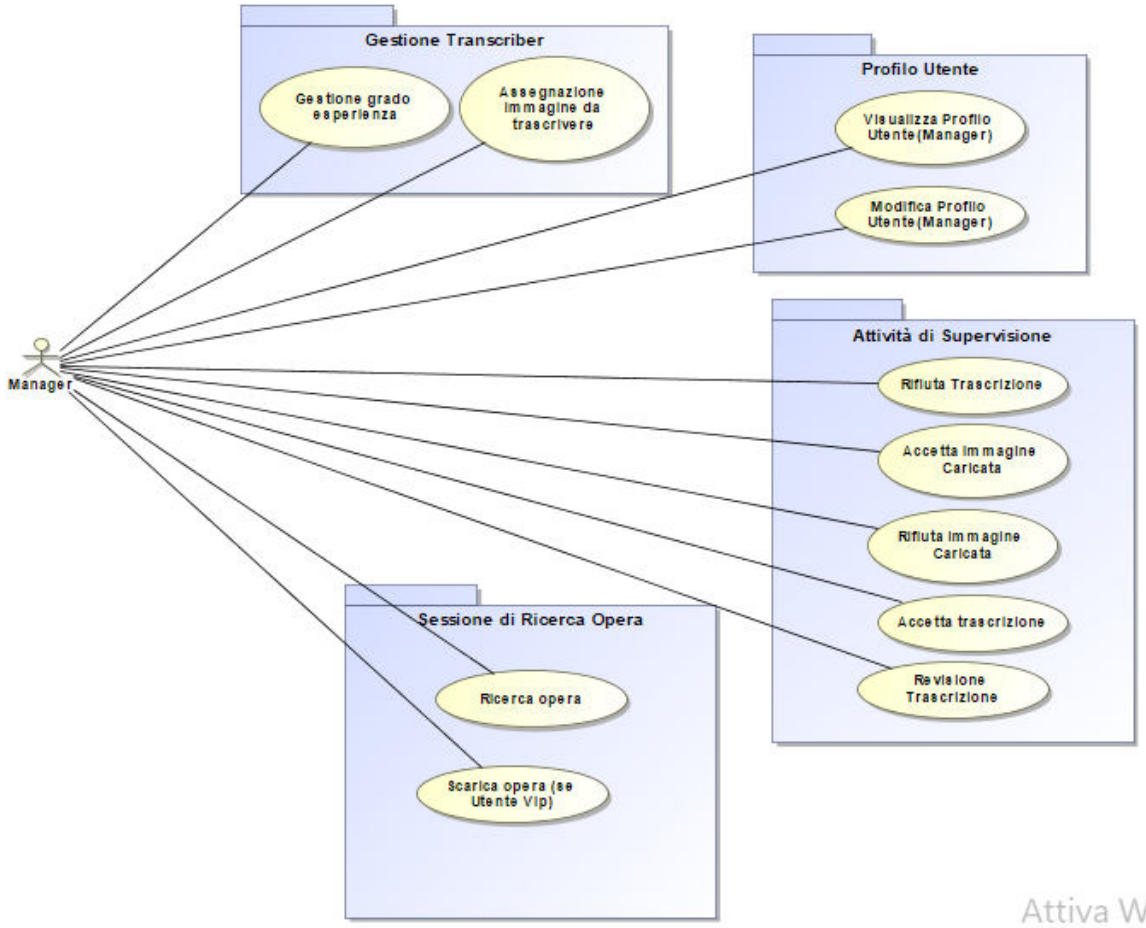
L’utente, dopo essersi iscritto nel sistema, può visualizzare il suo profilo, diventare transcriber, e può modificare il suo profilo (riportato nel package “**Profilo Utente**”). Inoltre può effettuare una ricerca dell’opera, e se è un **Utente Vip**, può scaricare l’opera (riportato nel package “**Sessione di ricerca opera**”).

Use case per il transcriber:



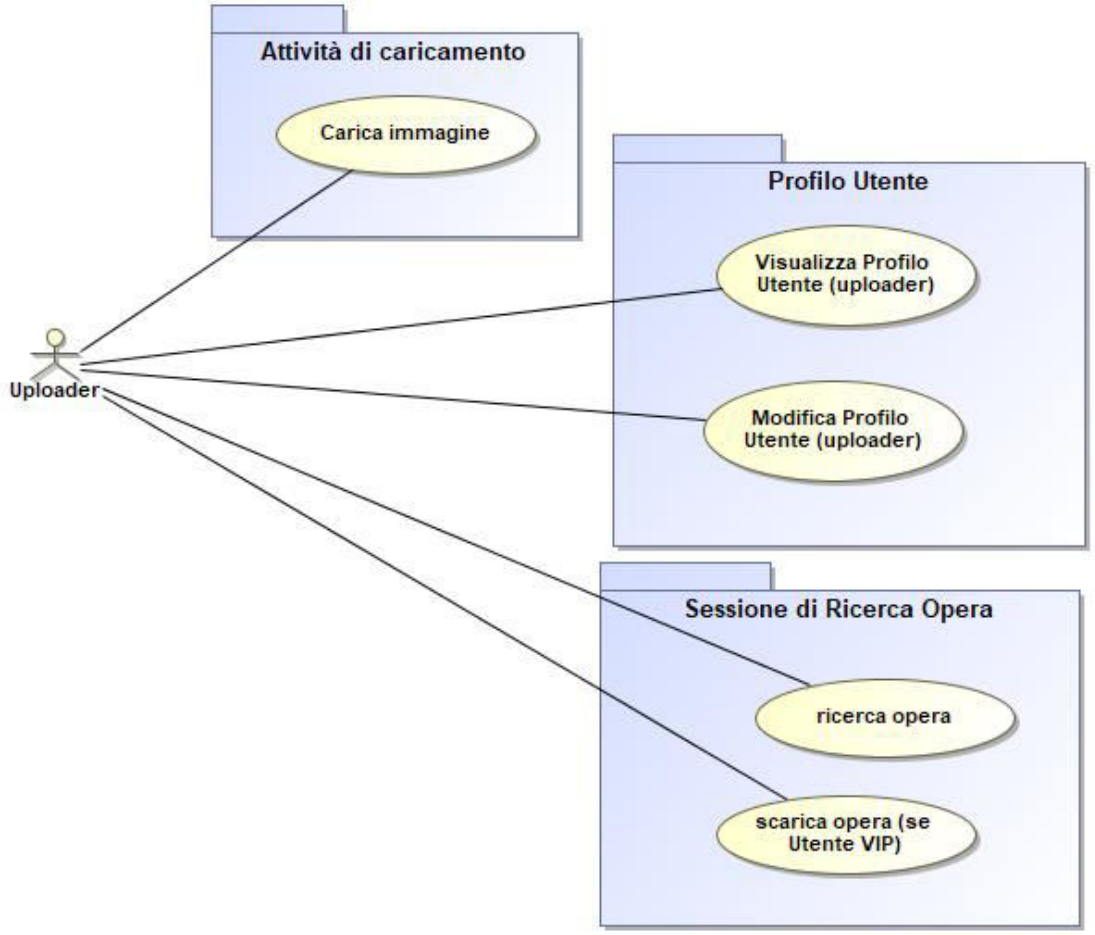
Un utente registrato che è anche transcriber dopo aver compilato l’apposito modulo, può svolgere le stesse azioni di un normale **utente loggato**, inoltre svolge le azioni di un transcriber, ossia trascrivere opere ed aumentare il suo grado di esperienza (quest’ultimo aspetto è gestito, in particolare, dal manager).

Use case per il manager:



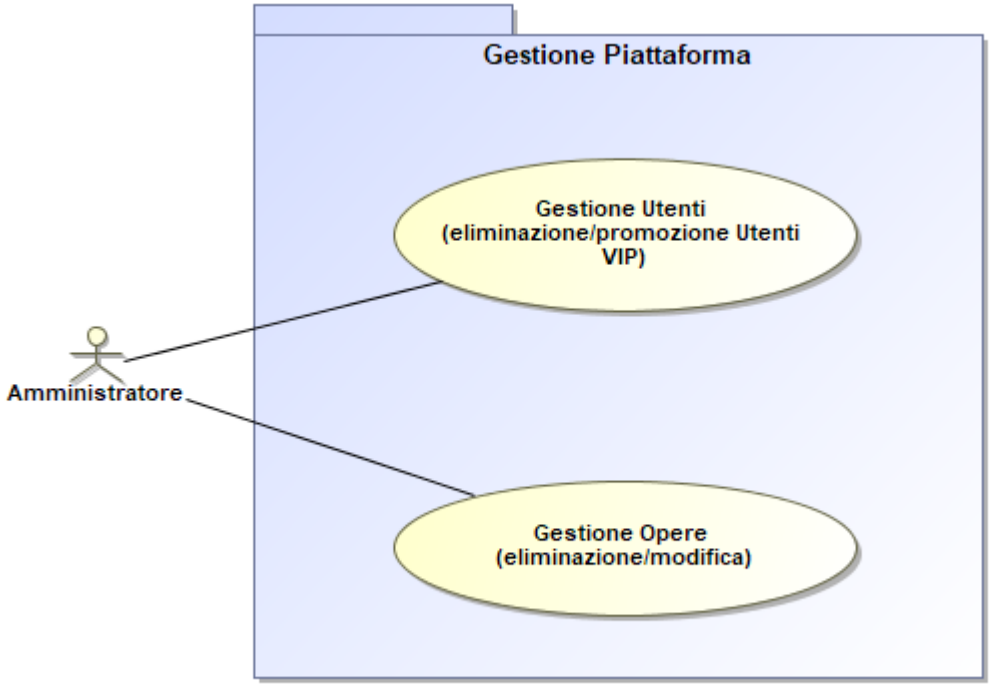
Un manager può svolgere le stesse funzionalità di un utente loggato. Inoltre egli svolge attività si supervisione come quelle riportate in figura. Infine egli assegna il lavoro ai transcriber, aumentandone, in caso, il grado di esperienza.

Use case per l’uploader:



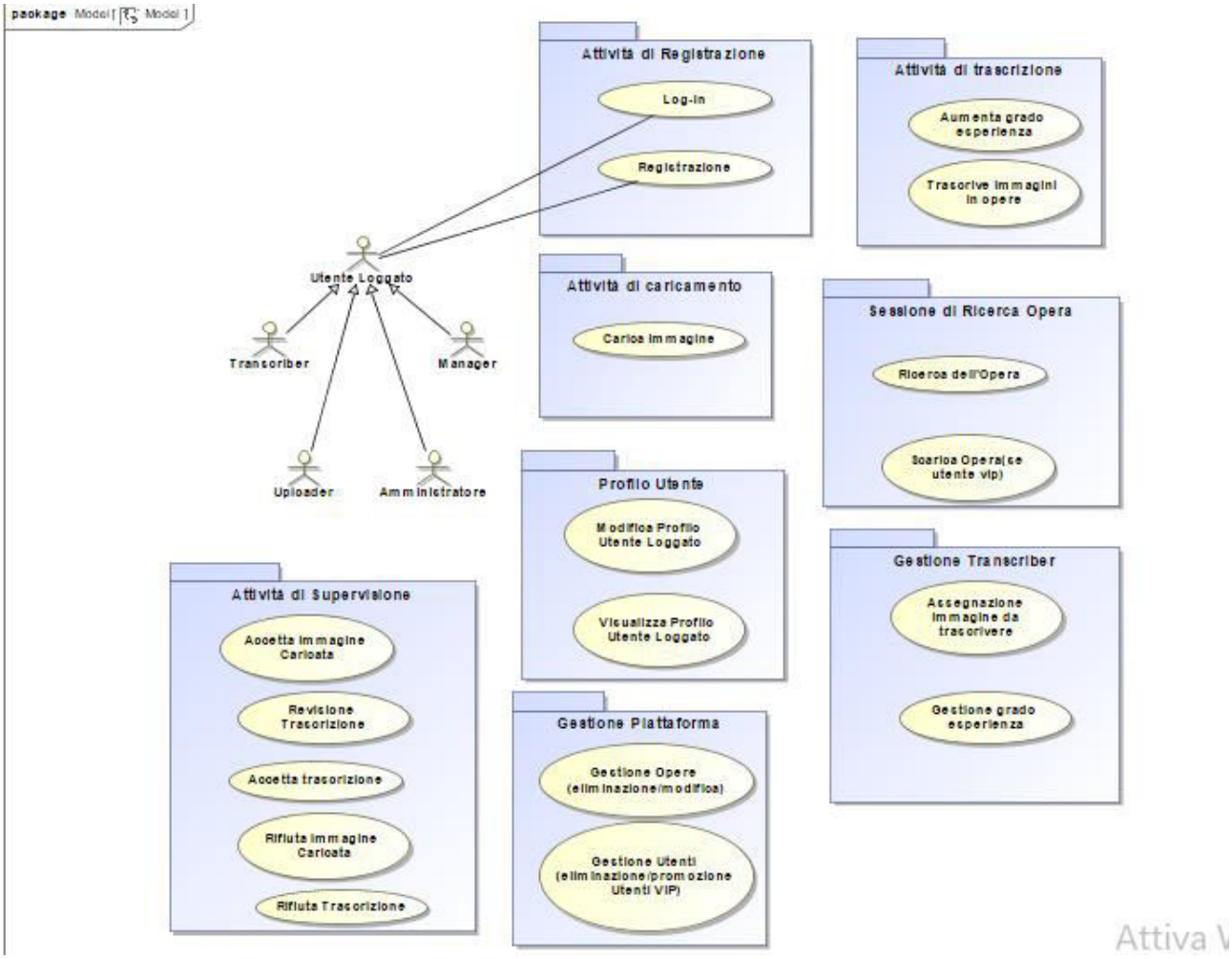
Un uploader svolge le stesse azioni di utente. In più egli ha il compito di caricare le immagini delle singole pagine di un’opera.

Use case per l’amministratore:



L’amministratore si può considerare come l’attore più importante del sistema. Egli svolge attività di gestione dell’intera piattaforma e, in particolare, si occupa dell’eliminazione e della modifica di opere. Inoltre ha la possibilità di gestire gli utenti e di promuoverli ad utenti vip, i quali hanno possibilità di scaricare immagini dell’opera che desiderano.

Use-case completo:



Lo use-case completo comprende tutti gli use-case mostrati in precedenza.
N.B: Si noti che le linee di associazione tra attore e use-case non sono state messe di proposito per non “sporcare” il disegno e renderlo di difficile comprensione. La descrizione dello use-case completo sarà espressa negli Scenari.

1.5) Scenari:

Attività di Registrazione: La registrazione nella piattaforma è svolta da tutti gli attori presenti nel sistema, inserendo opportuni dati consoni ad una corretta registrazione (anagrafica e altro) e scegliendo, inoltre, uno username ed una password. Si noti che, come descritto nei requisiti funzionali, si è deciso di scegliere nel sistema solo utenti che abbiano nickname che li identifichino univocamente, ciò significa che non possono esserci utenti con nickname uguali.

Profilo Utente: Tutti gli utenti avranno un loro profilo utente, il quale consentirà loro di visualizzare le proprie generalità e i propri ruoli all’interno della piattaforma. La sezione profilo utente permette agli stessi anche di modificare il proprio profilo.

Attività di Trascrizione: L’attività di Trascrizione, da come si può evincere, è svolta dal transcriber, il quale si occupa di trascrivere immagini in opere.

Attività di caricamento: L’attività di caricamento è svolta dall’uploader, il quale ha il solo compito di caricare l’immagine di una pagina dell’opera di appartenenza.

Sessione di Ricerca Opera: Questa attività può essere svolta da tutti gli utenti del sistema, e consiste nel fornire un servizio di ricerca dell’opera che si desidera (se presente nel sistema).

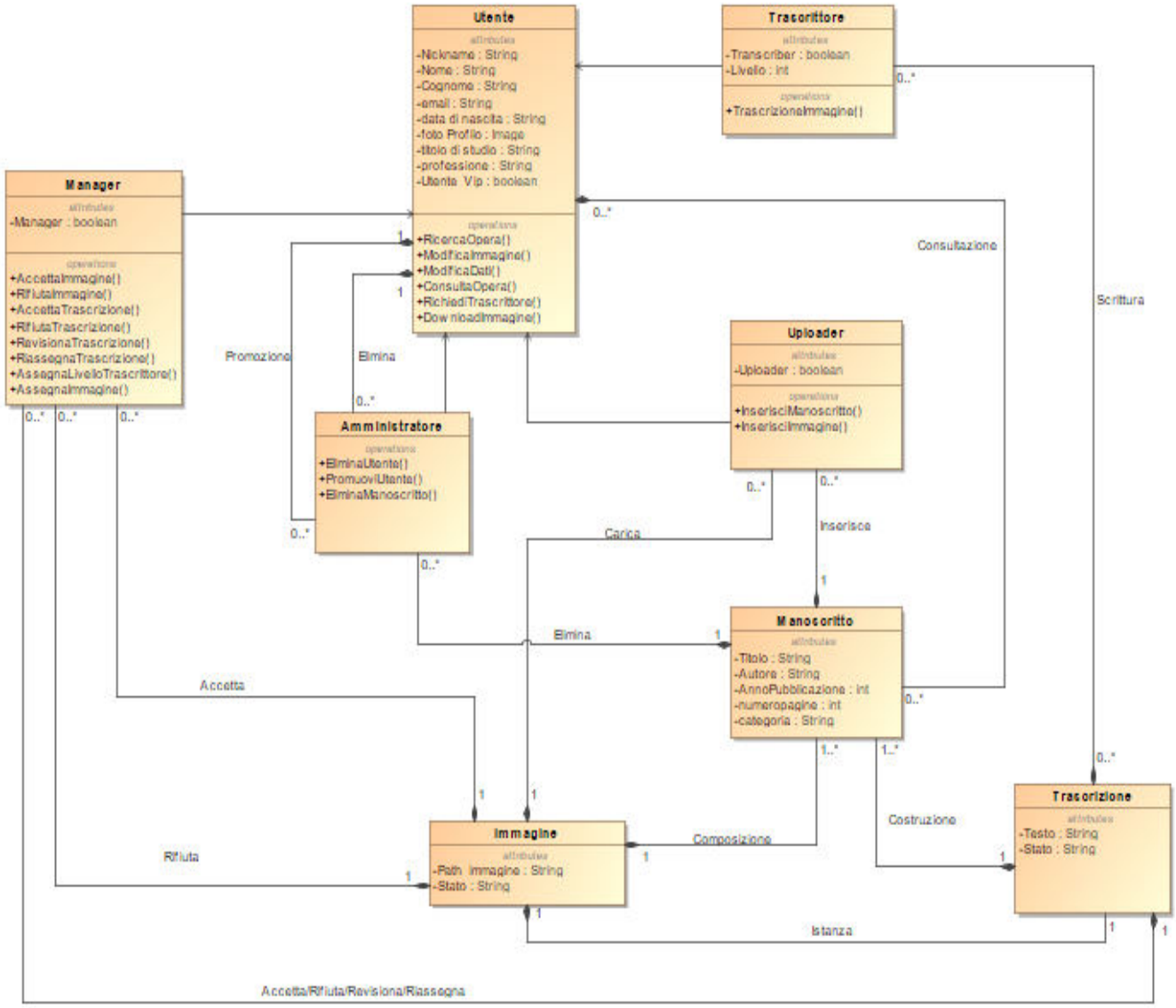
Gestione Transcriber: Questa attività è svolta dal manager, e consiste nell’assegnare un’opera da trascrivere ad un transcriber e gestirne il suo grado di esperienza.

Attività di Supervisione: L’attività di supervisione è svolta sempre dal manager, e consiste nella supervisione dei vari lavori che lo stesso manager ha assegnato ad uploader e transcriber, come ad esempio l’accettazione, il rifiuto o la modifica di una trascrizione o il declinamento di un’immagine caricata dall’uploader, che, ad esempio, non è di sufficiente qualità.

Gestione Piattaforma: La gestione della piattaforma è svolta dall’amministratore del sistema, che come già accennato, può essere considerato come l’attore più importante all’interno della piattaforma stessa. Infatti egli ha pieno controllo del sistema, infatti è in grado di modificare/cancellare il profilo di un utente o di un’ opera.

1.6) **Modello di Dominio**:

Per modello di dominio si intende un insieme di classi utili a rappresentare e finalizzare i problemi che si possono presentare nella costruzione di un sistema, come ad esempio nel nostro caso:



Nel seguente modello vengono riportate tutte le classi scelte che si identificano all’interno del sistema, riportando, per ognuna di esse, i vari attributi e i vari metodi. Nel modello presentato sono presenti opportune relazioni tra le varie classi, in cui in più, sono riportate le varie associazioni. Sono presenti anche le opportune generalizzazioni tra le varie classi, come ad esempio i vari tipi di utente che ci possono essere all’interno del sistema, generalizzati, appunto, dalla classe utente. Nel seguente dominio sono anche state messe relazioni di composizione, come ad esempio tra la classe manoscritto e la classe pagina. Una relazione di composizione, nell’esempio, identifica l’esistenza obbligatoria di un manoscritto affinchè esistano una o più pagine, altresì, se non vi è alcun manoscritto, non possono esistere pagine ad esso associate.

1.8) Individuazione di classi entity, boundary e controller:

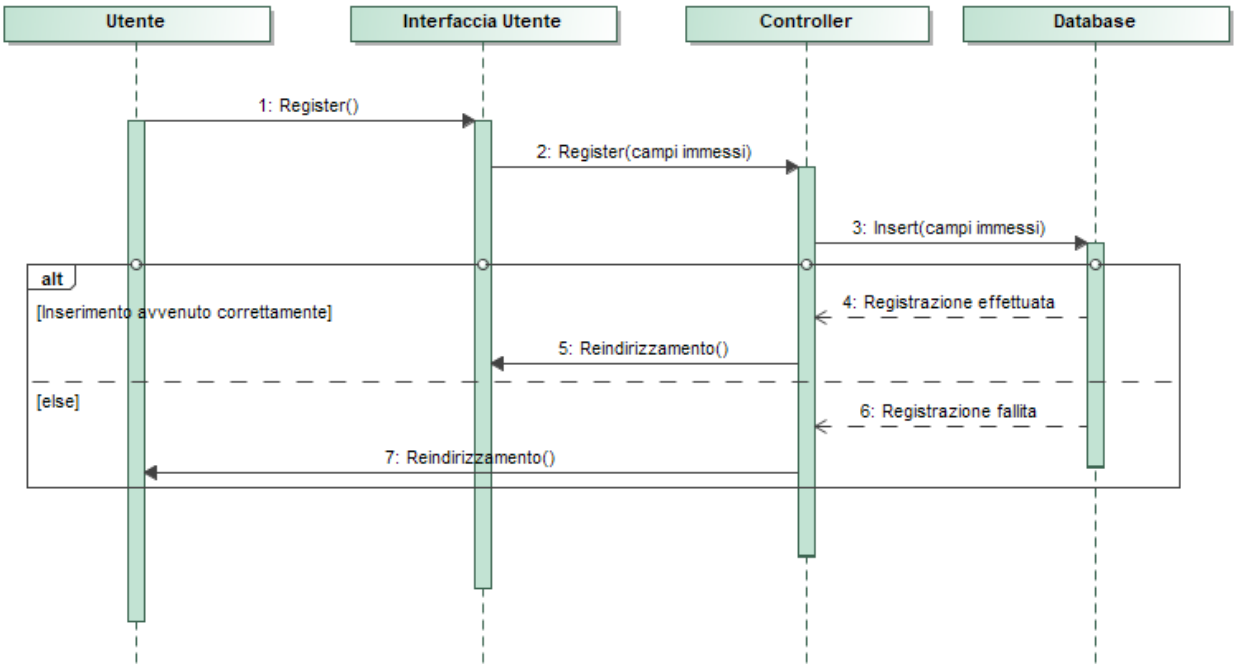
Le classi entity del sistema possono essere considerate le stesse presenti all’interno del modello di dominio. Sarà presente una classe **Utente**, la quale conterrà i vari attributi e operazioni che caratterizzano l’utente stesso. Le classi **Utente vip, trascrittore, uploader, amministratore e manager** sono sottoclassi della classe principale Utente, quindi tutte le sottoclassi citate erediteranno gli stessi attributi e operazioni della classe padre utente. In aggiunta, le sottoclassi avranno un attributo boolean, il quale assumerà valori true o false in base al ruolo che gli utenti registrati svolgono all’interno della piattaforma. Le altre classi entity scelte sono:

Manoscritto: Questa classe rappresenta, appunto, un manoscritto all’interno della piattaforma. Essa avrà i vari attributi che identificano lo stesso manoscritto, come l’autore del manoscritto, il titolo del manoscritto, l’anno di pubblicazione, numero di pagine al suo interno, ecc... Inoltre esso avrà un campo id al suo interno, il quale lo identificherà univocamente all’interno del sistema.

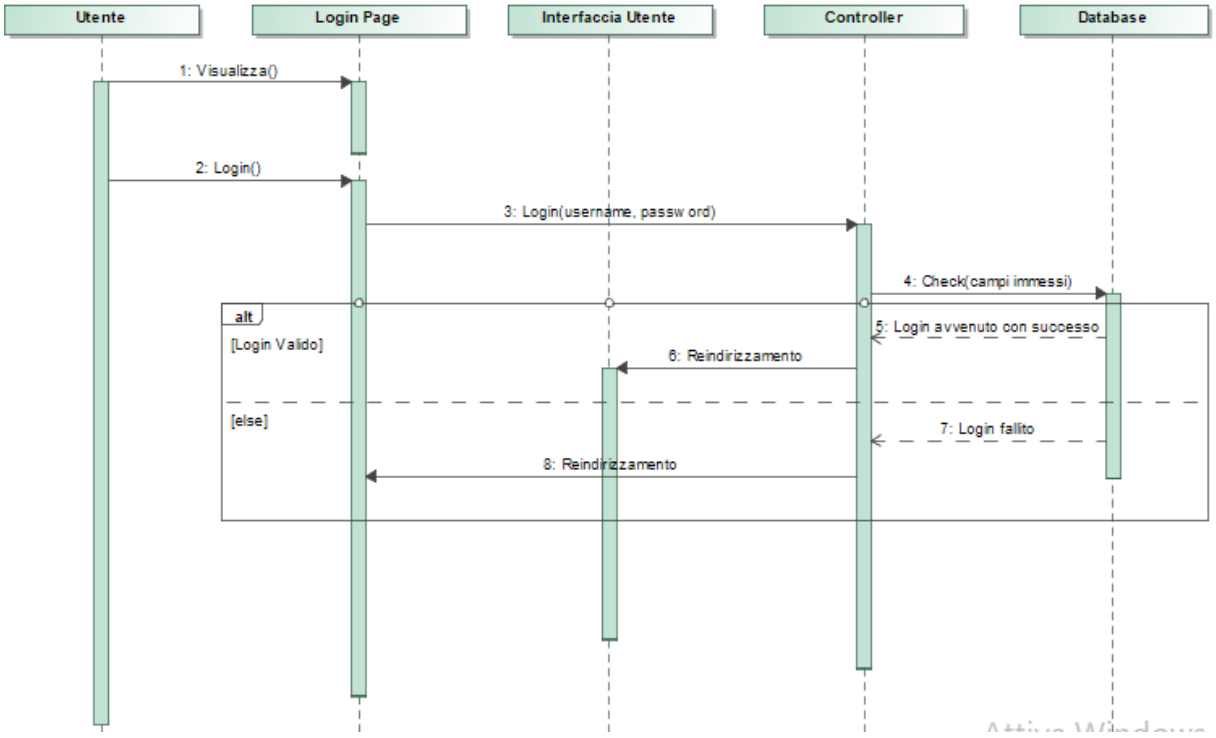
Trascrizione: Questa classe identifica il testo della trascrizione relativa ad una immagine, è quindi opportuno che essa abbia un campo testo.

Immagine: La classe Immagine identifica una certa immagine di una pagina di un’opera all’interno della piattaforma. Ad una immagine possono essere associati un attributo qualità, in base al quale il manager può declinare il caricamento dell’immagine stessa, ed un attributo che identifica una variabile image, la quale servirà a contenere l’immagine vera e propria.

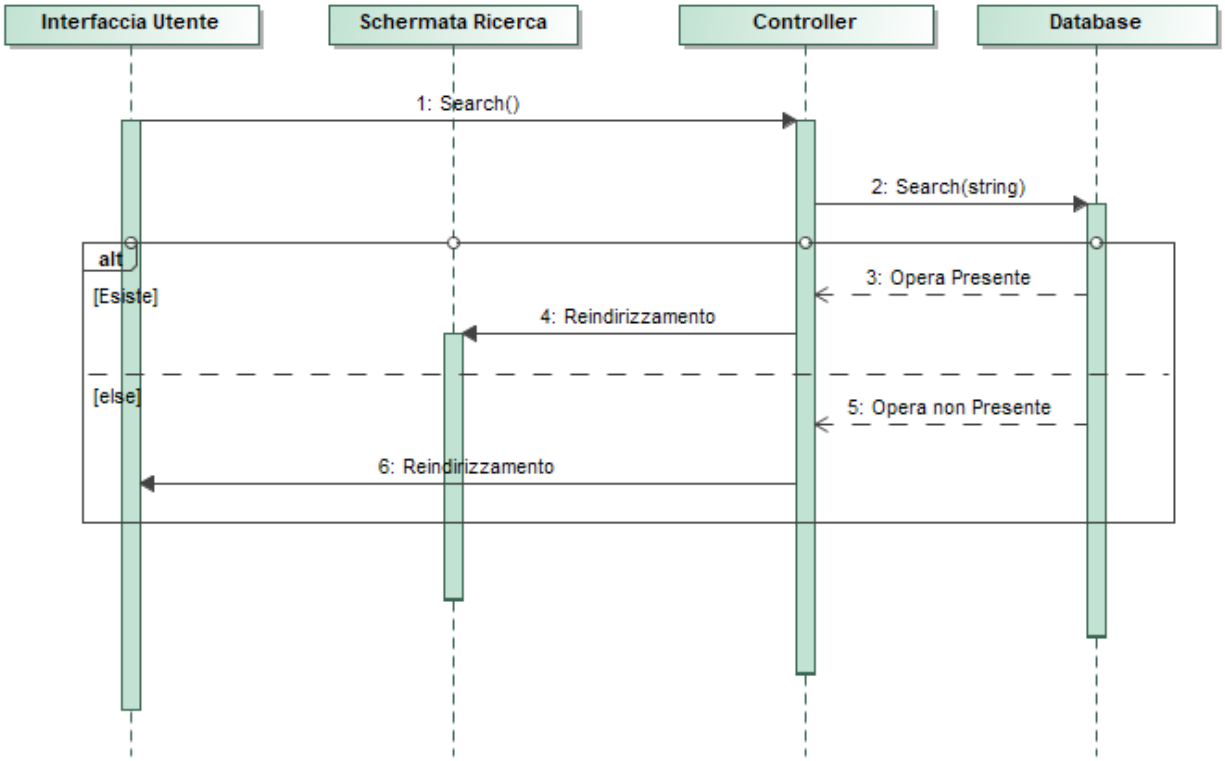
Le classi citate saranno opportunatamente gestite e mantenute in un Database. Le classi boundary identificheranno le varie interfacce che faranno da tramite tra il sistema e l’utente, come ad esempio, finestre, popup di avviso e menù. Il tutto sarà all’interno di un unico package GUI (Graphical User Interface). Le classi Controller si occuperanno del corretto controllo del flusso di informazioni tra le classi boundary e le classi entity nel sistema. Quindi ad ogni classe boundary del package GUI verrà assegnata una classe Controller del package Controller.
Una rappresentazione del funzionamento di tali classi può essere rappresentato nei **Sequence Diagram** illustrati di seguito.



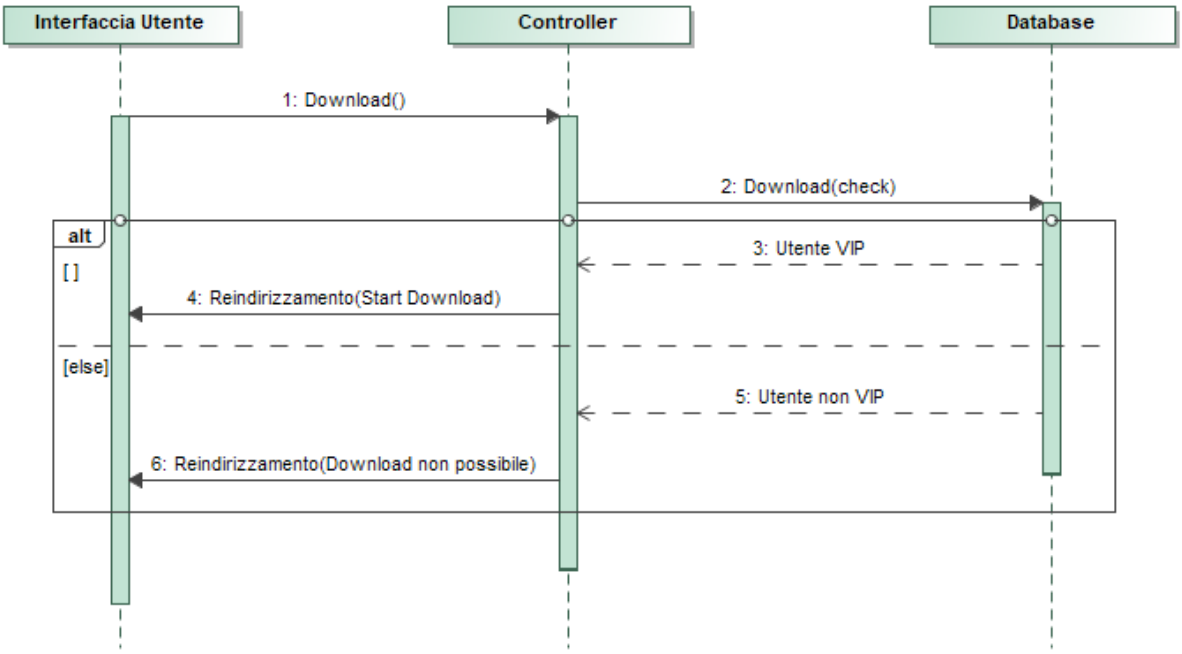
1-Registrazione Utente.



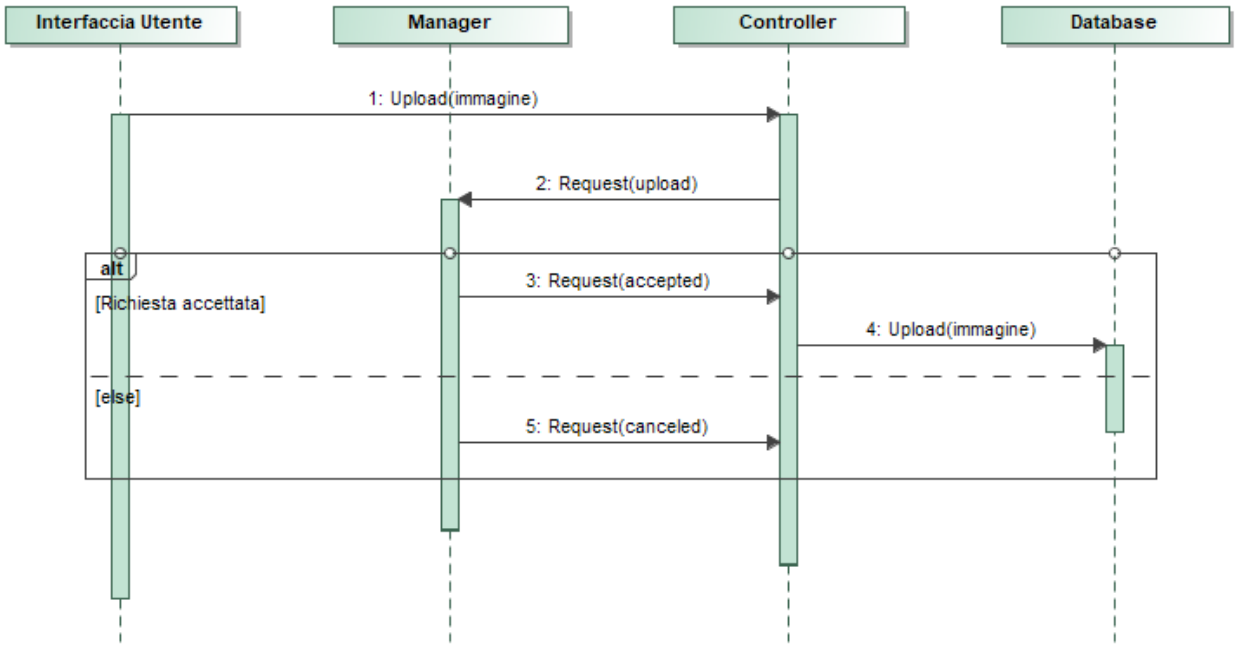
2-Login Utente



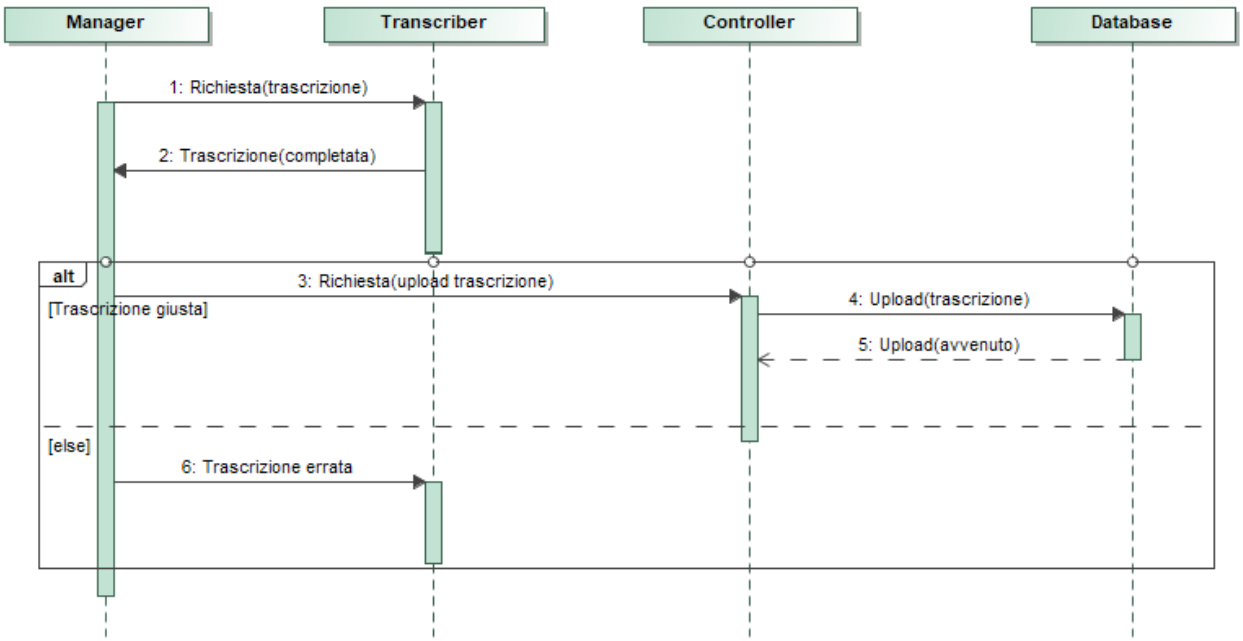
3-Ricerca Opera



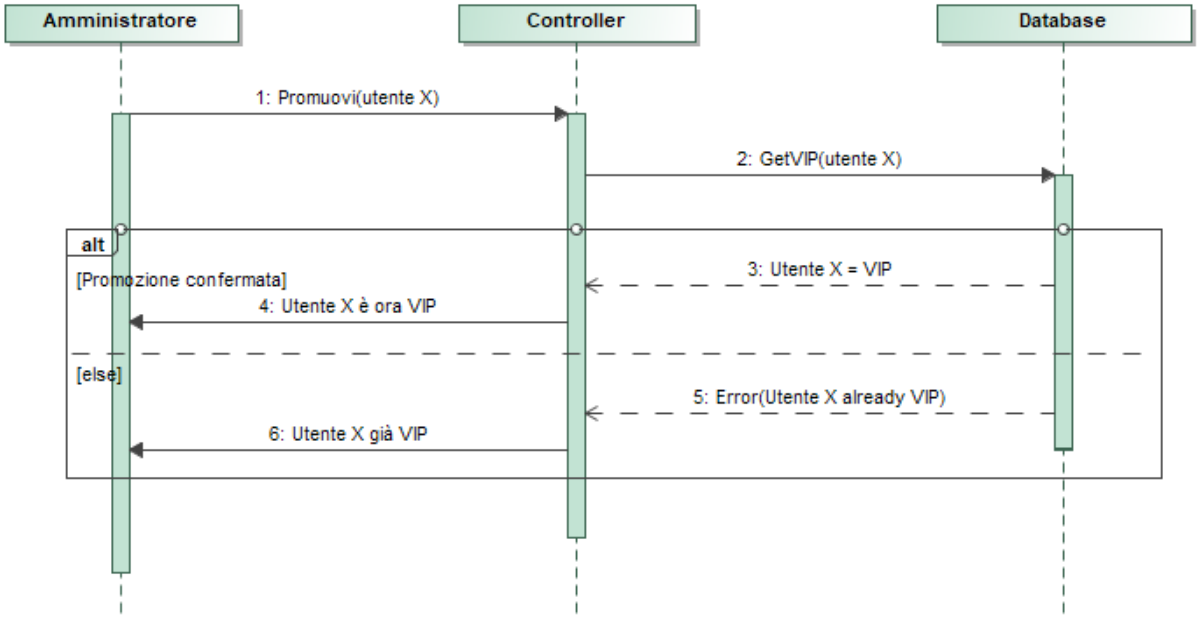
4-Download Opera.



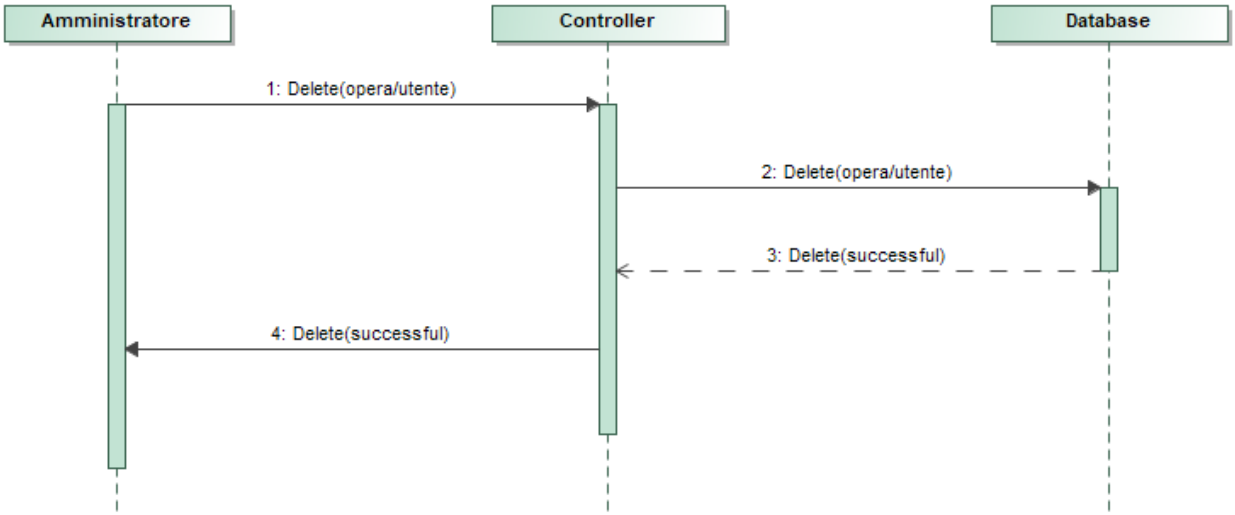
5-Upload immagine



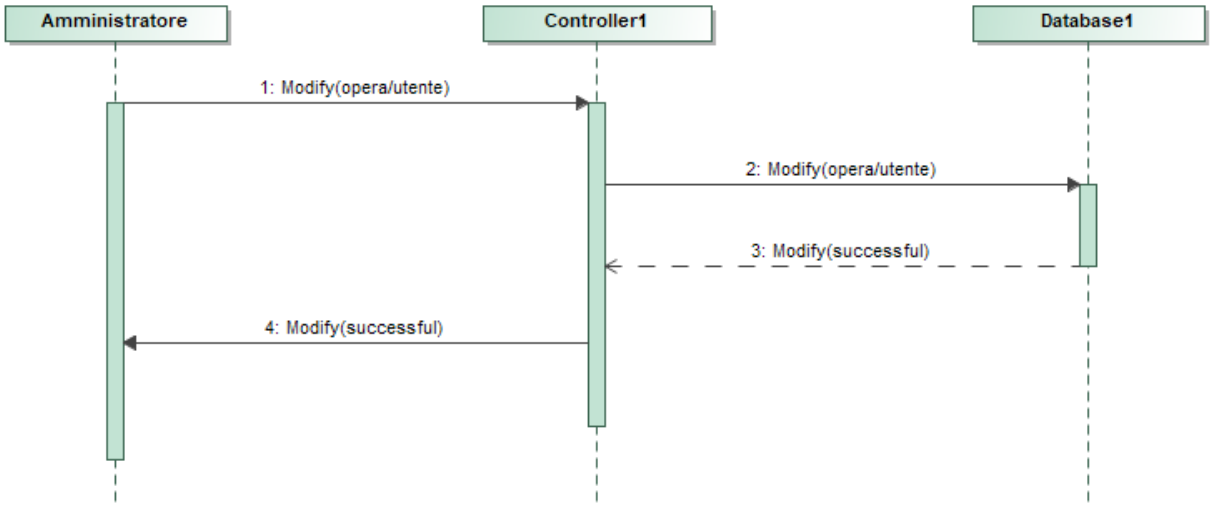
6- Caricamento Trascrizione.



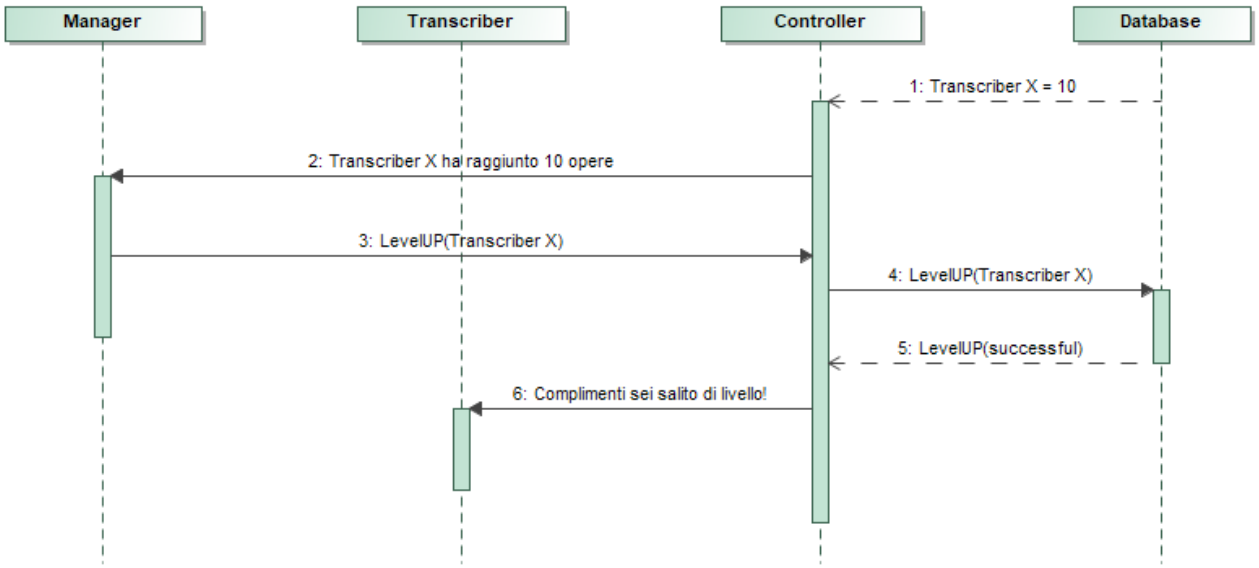
7- Promozione Utente VIP



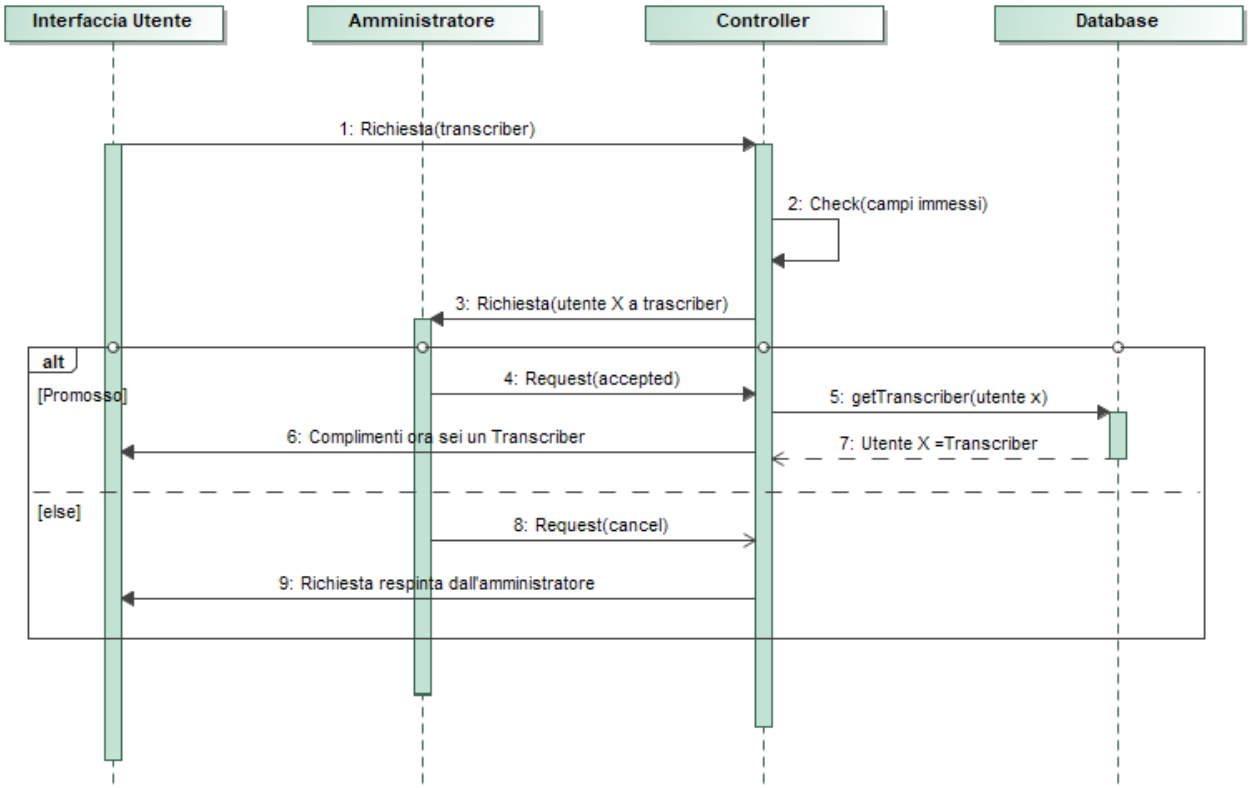
8- Eliminazione opera/utente.



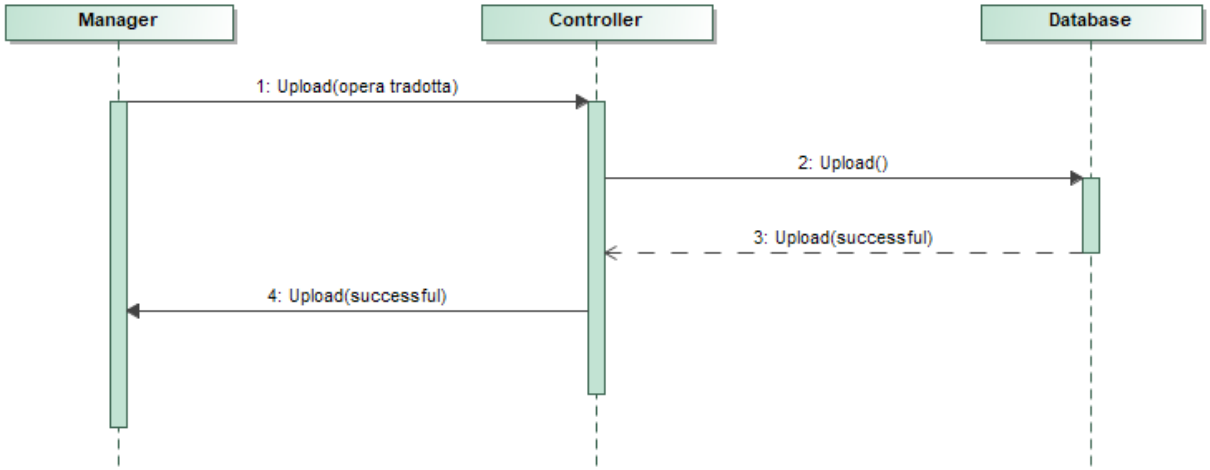
9- Modifica opera/utente.



10- Avanzamento livello esperienza Transcriber.



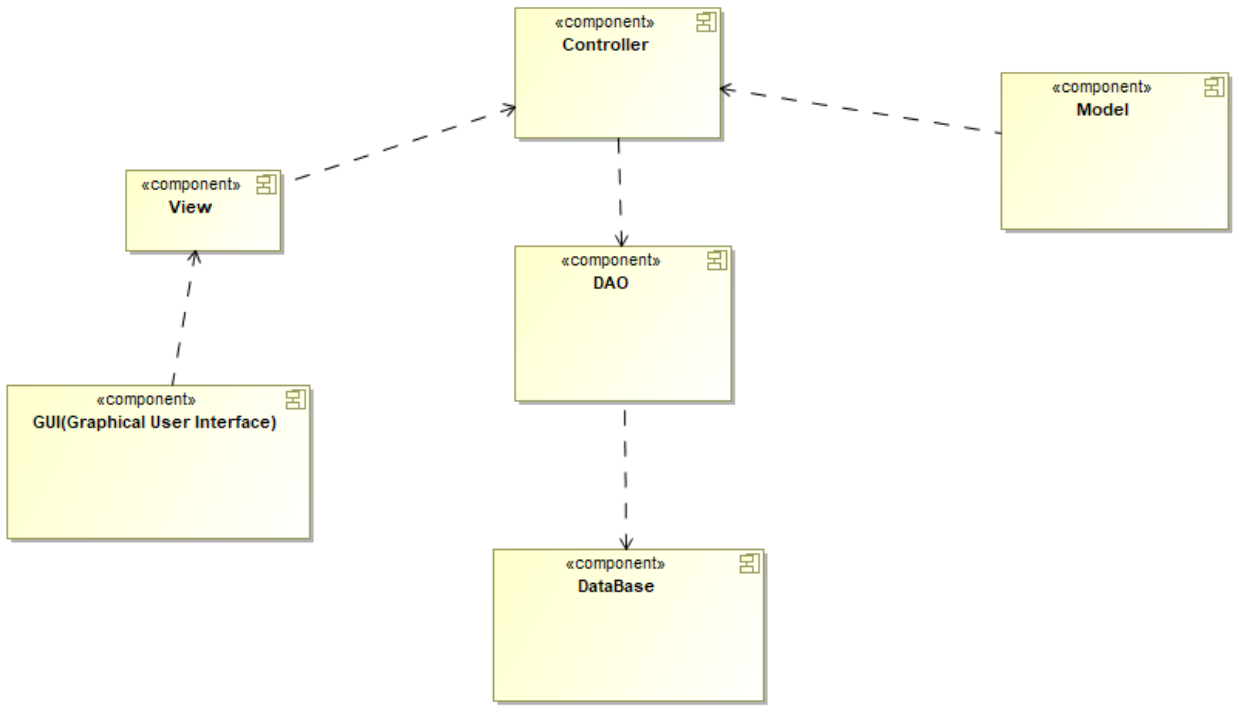
11-Diventa Transcriber



12-Pubblicazione Opera.

2.1) System Design

Per la descrizione dell’architettura software del sistema si è scelto di usare un **Component Diagram**:



2.2) Descrizione dell’architettura:

La componente **GUI (Graphical User Interface)**, come già descritto, identifica le varie interfacce che faranno da tramite tra il sistema e l’utente. La componente **View** sarà identificata da un altro package, e più approfonditamente la view verrà implementata in ogni entità della GUI e verrà chiamata in campo per un primo controllo sull’input utente, come previsto dal pattern **MVC (Model View Controller)**. Quindi, ad esempio, la classe profiloUtente, implementerà utente e controllerProfiloUtente. In particolare, la componente View sarà utilizzata dal controller per un corretto utilizzo dei dati e una corretta gestione di essi da parte dello stesso utente attraverso il costrutto ***implements***. Il tutto sarà gestito in modo da essere di facile apprendimento e user-friendly (vedi requisito non funzionale **Usability**). La già citata componente **Controller** si occuperà del corretto flusso di dati immessi dall’utente, effettuando, nel caso, re-indirizzamenti e gestione delle funzionalità nel sistema, come già dimostrato nei vari **Sequence Diagram.** La componente **Model** avrà al suo interno tutti i vari oggetti utilizzati dalle altre classi, ossia quelle già descritte e specificate nel **Domain Model**. Questo avviene mediante funzioni che hanno il compito di interagire con gli oggetti selezionati. Per esempio, quando si seleziona da una lista un oggetto e si avvia, per esempio, una certa funzione “cancella oggetto”, attraverso tale funzione si va a interagire con la classe in questione eliminandola dal contesto. Questo flusso di azioni è gestito dalla componente associata alla componente Model, il Controller. L’altra componente che incontriamo è il **Data Access Object (DAO)**. Il

DAO è la frontiera più esterna che separa il codice Java dall’effettiva e fisica componente del sistema **Database**, e tramite l’implementazione di alcune librerie specifiche di Java sarà possibile la corretta comunicazione tra Java e MySQL.

2.3) Descrizione delle scelte e strategie adottate:

Il problema principale da esporre è la gestione e l’ITER-procedurale dei dati immessi dall’utente, questo perché si vuole garantire un accesso al Database pulito. Per fare ciò abbiamo identificato 4 sezioni distinte:

1- Il **Model** rappresenta tutti i vari attori del sistema, per esempio utenteVip() extends utente() ecc... fanno parte delle entità su cui verranno fatte delle modifiche, si interagirà, e in generale saranno partecipe di varie azioni.

2- La **GUI** sarà l’unica interfaccia con cui l’utente si relazionerà direttamente. Attraverso l’implementazione delle classi GUI con classi di tipo Controller avverrà la comunicazione dei dati tra le prime con le seconde. In più tale classe per raccogliere o settare i vari dati dovrà necessariamente implementare la classe Model con cui interagisce ed ereditarne i relativi metodi di settaggio e di return dei vari attributi.

3- La **View** avrà il compito di controllare il primo input fatto dall’utente, come già descritto sopra. Tale controllo sarà possibile grazie all’implementazione di tale classe nelle singole classi GUI. Per fare ciò la view avrà varie funzioni che verranno richiamate nel momento del settaggio degli attributi, se l’esito del controllo dell’input è positivo la view invierà tale input al Controller.

4- Il **Controller** fa da prima linea per input errati nella forma (es. caratteri non ammessi/errati). In tale evenienza si rida il controllo alla GUI attraverso la view che ne evidenzia a video l’errore riscontrato. Caso in cui il primo check dell’input viene accettato si comunicano tali attributi alla componente DAO per accertare, per esempio, la già presente esistenza di un eventuale nickname all’interno del Database. Quindi, effettivamente il controller ha un ruolo di prima scrematura e controllo dei dati.

5- Il **DAO** è un’interfaccia di Java che permette un corretto e più semplice utilizzo del Database fisico, qualunque esso sia. Il DAO riceve dati dal Controller, e su questi dati effettua query passandole alla gestione del Database che si sceglie.

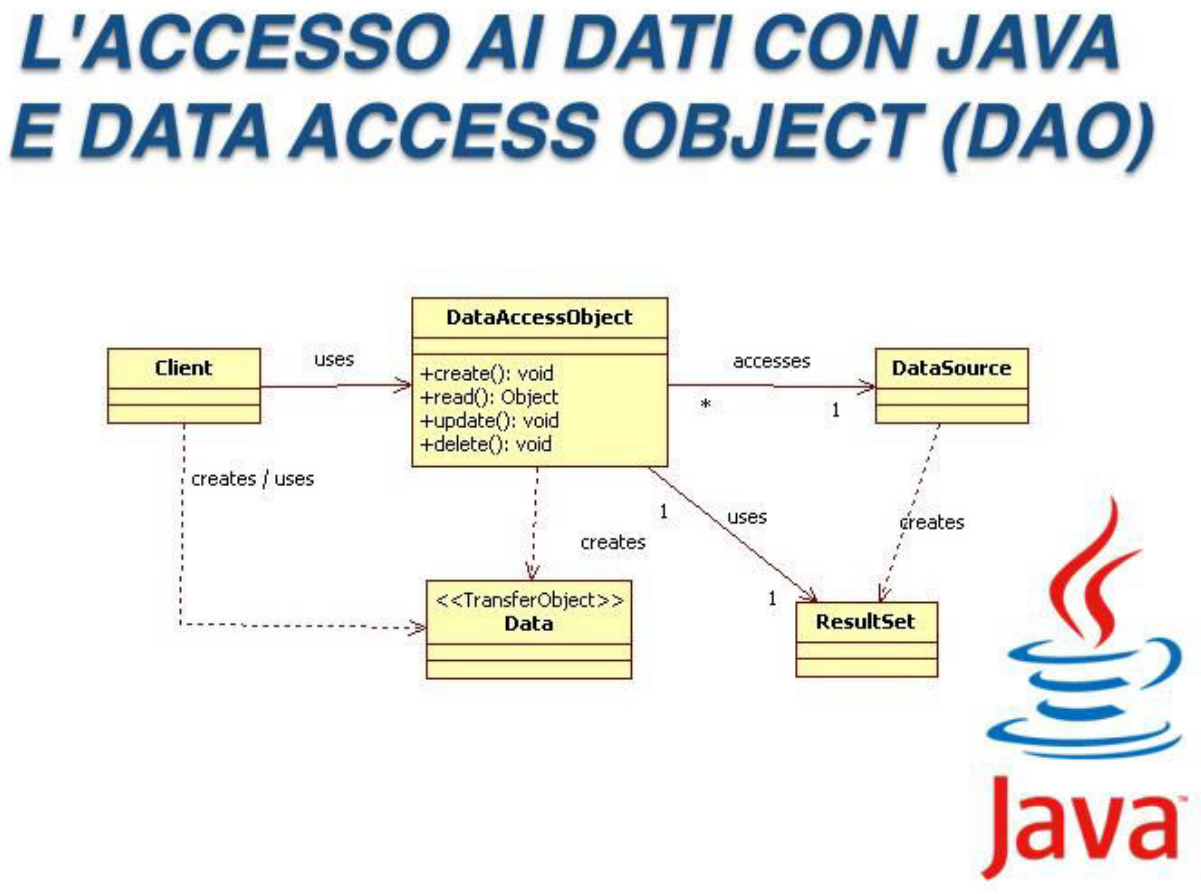
6- Il **Database** da noi utilizzato sarà MySQL. Esso garantirà persistenza, affidabilità e integrità dei dati relativi alle varie entità nel sistema.

2.4) Design Patterns:

Data-Access-Object:

Si è scelto di usare come design pattern il DAO (**D**ata **A**ccess **O**bject) per gestire, in particolare, la persistenza dei dati nel Database. Il pattern si basa su:

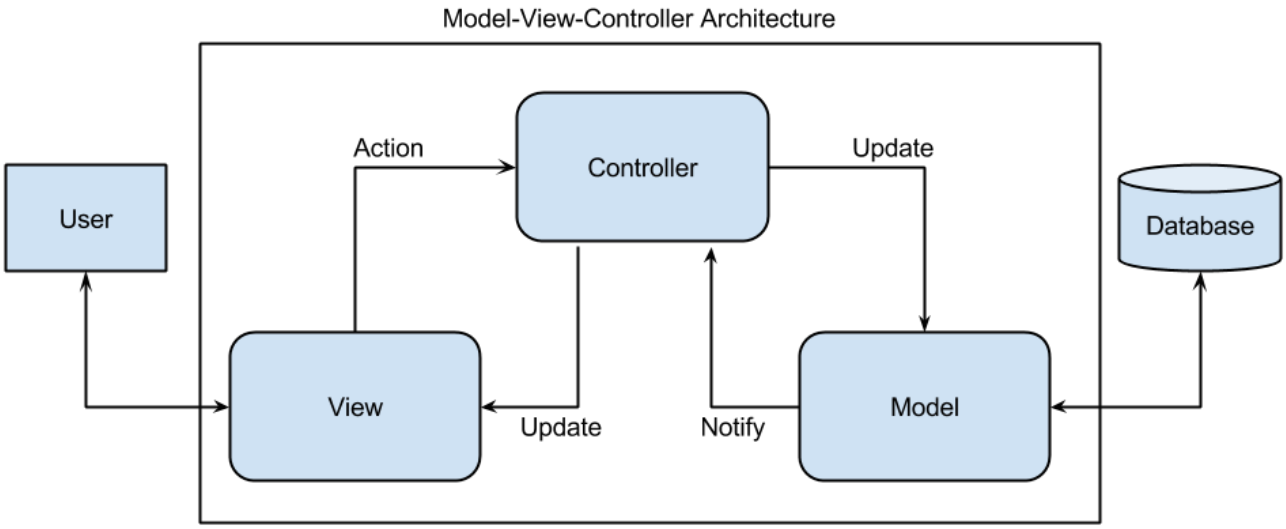
1- **DAO Interface**: Un’ interfaccia che definisce la struttura generale della classe fisica ad essa associata. Essa specifica semplicemente il nome del metodo, il tipo da restituire e i parametri da utilizzare senza specificarne il corpo.
2- **DAO Classe Concreta:** Si intende una classe che implementa l’interfaccia ad essa associata, ciò vuol dire che dovrà obbligatoriamente contenere i metodi specificati nella sua interfaccia importata specificandone però il corpo del metodo. Tale classe ha il compito di estrapolare dati dal database per poi passarli al **Controller**.



Model-View-Controller:

Questo pattern è basato sulla separazione di 3 componenti del pattern:

1-Model
2-View
3-Controller



Model: Il Model ci permette di identificare le funzionalità della piattaforma, in specifico attraverso i metodi che sono racchiusi in ogni classe Model andiamo a interagire con i vari soggetti della piattaforma, andandone a manipolare i dati. Il Model è una componente molto importante, in quanto interagisce direttamente con il Database e viceversa. Tale relazione permette la permanenza dei dati, grazie soprattutto al Database. Ogni volta che si deve manipolare un soggetto dal Database si estrapolano informazioni, e con tali informazioni si va a creare un’entità Model. Viceversa quando si modifica un’entità Model per garantire persistenza dati si aggiorna il Database.

View: La View si occupa di gestire ogni evento all’interno della piattaforma. E’ il package più denso di funzionalità in quanto va a gestire tutte gli eventi della GUI, in particolar modo, ogni volta che l’utente interagisce con la piattaforma, la View ha il compito di mostrare i dati sotto forma di immagini e fa una elaborazione iniziale dei dati che l’utente immette. In un certo senso si può dire che la View fa da tramite tra l’utente e la piattaforma stessa.

Controller: Il controller è un “ponte” che collega la View al Model, per poi finire nel DAO, il quale si occuperà dell’interazione con il Database per la gestione delle varie richieste.
Si occupa anche di controllare che tutto ciò che l’utente immetta sia corretto, in modo tale da garantire una corretta interazione con il Database.

Utilizzando il pattern **MVC**, mantenendo la separazione in 3 stati, è possibile fare una distinzione tra i vari concetti.

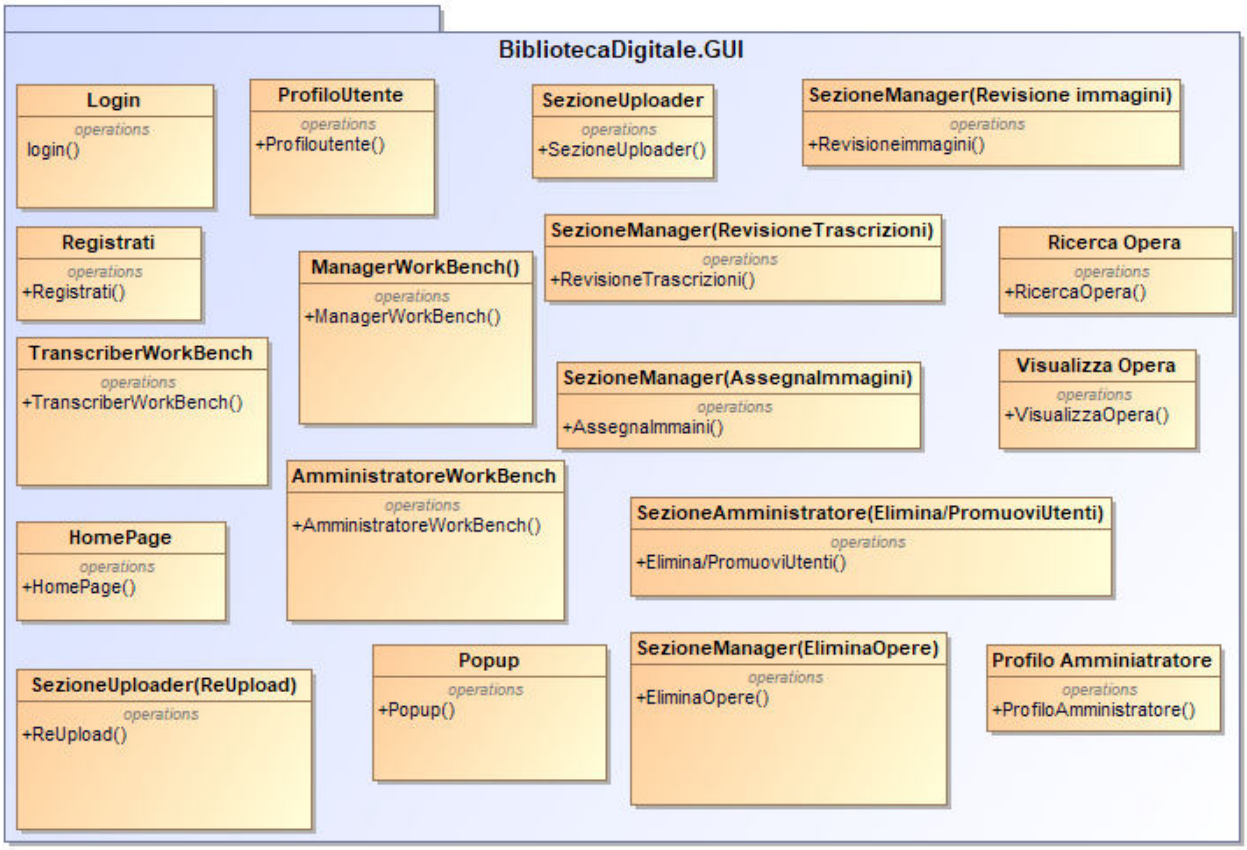
3.0) Software Design:

La piattaforma realizzata è stata separate in diversi package, con lo scopo di mantenere separati i dati presenti nelle varie classi, garantendone una separazione logica. I package individuati sono i seguenti:

- View (contenente anche il package **GUI**);
- Controller;
- Model;
- DAO (contenete anche il package **Database**);

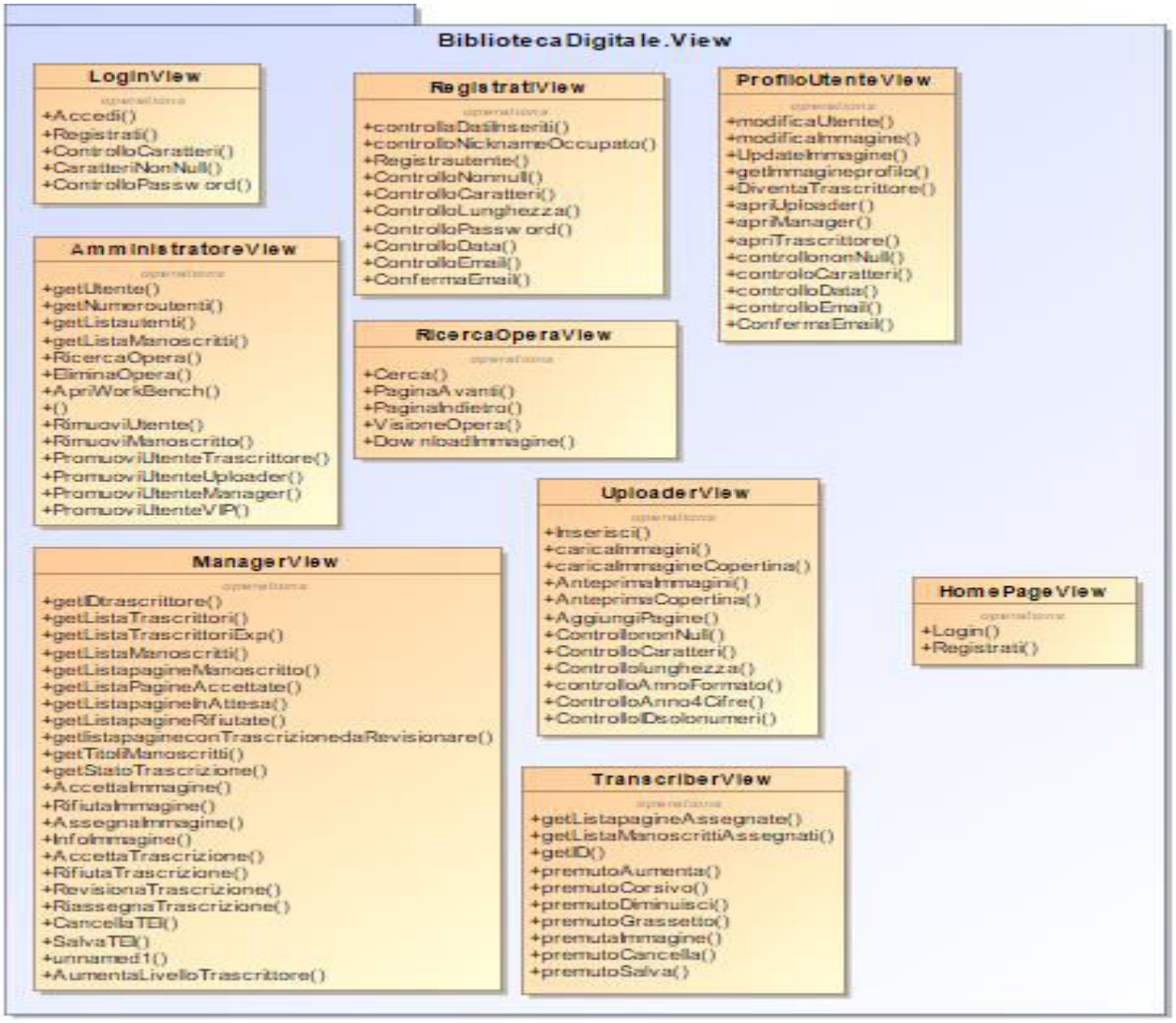
Di seguito sono stati riportati i **Class Diagram** di ciascun package, in cui ne verranno esplicitate le caratteristiche e le varie funzionalità.

Class Diagram – GUI:



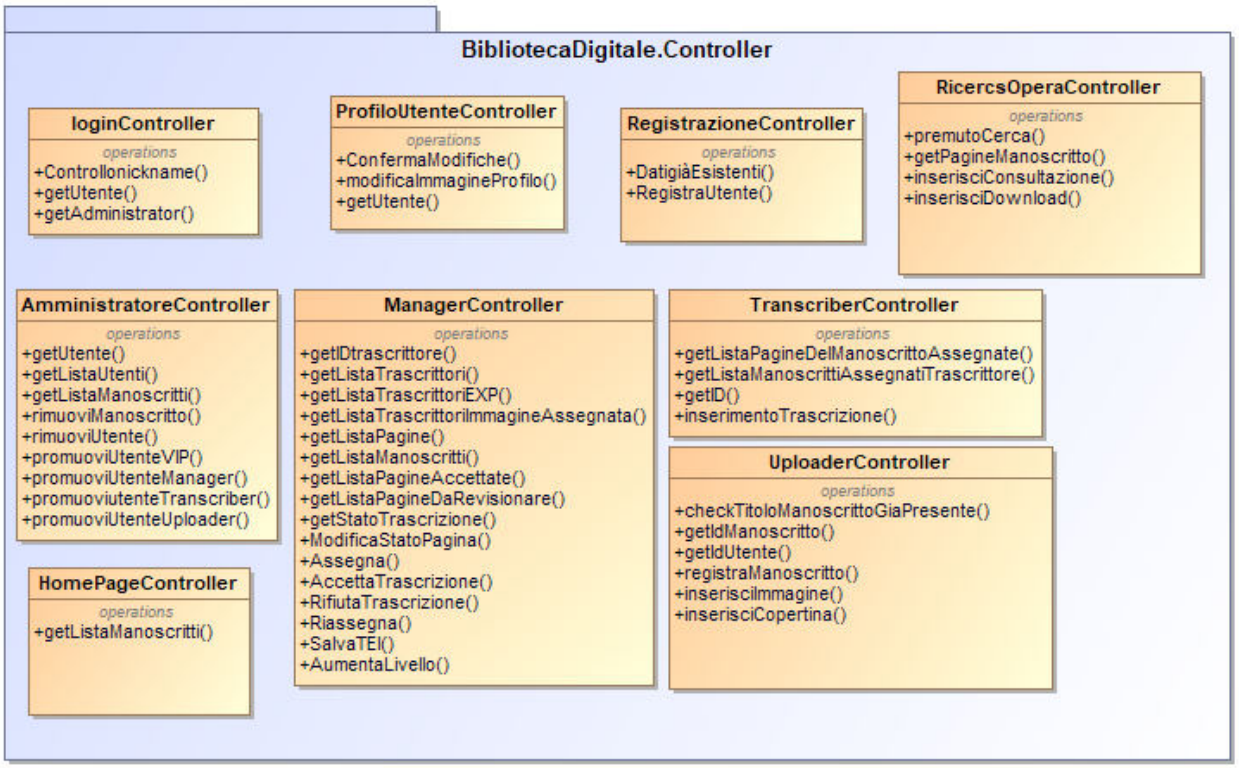
Questo package contiene tutte le GUI (Graphical User Interface) presenti all’interno della piattaforma. Le varie interfacce sono state costruite per mezzo del framework **Swing/AWT** di Java, garantendo un’interfaccia User-Friendly. Tra le GUI sono presenti anche delle finestre di “appoggio” che servono per rendere più facile il lavoro dei vari sottosistemi.

Class Diagram – View:



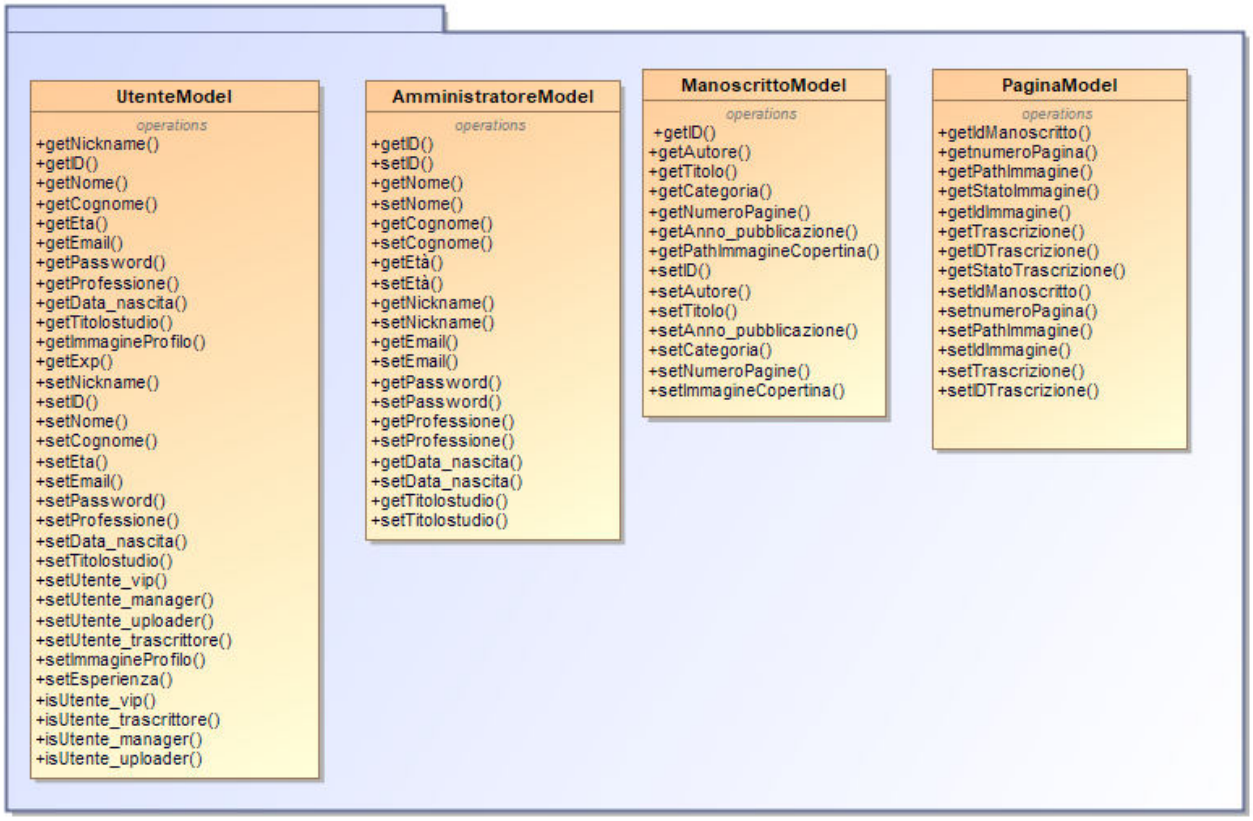
Il package GUI, creato al passo precedente, si avvarrà di metodi presenti nel package View. Una volta fatta questa operazione, la View avrà poi il compito di notificare l’avvenuta ricezione dell’input emesso dall’utente al package Controller, descritto in seguito.

Class Diagram – Controller:



Per ogni classe presente nel package View, avremo una classe corrispondente nel package Controller.

Class Diagram – Model:



Il package Model rappresenta le principali entità presenti nel sistema. Tali classi vengono rappresentate attraverso le principali caratteristiche e funzionalità che le contraddistinguono. All’interno di ogni Classe Model, inoltre, sono presenti costruttori, ed i rispettivi metodi getter/setter.

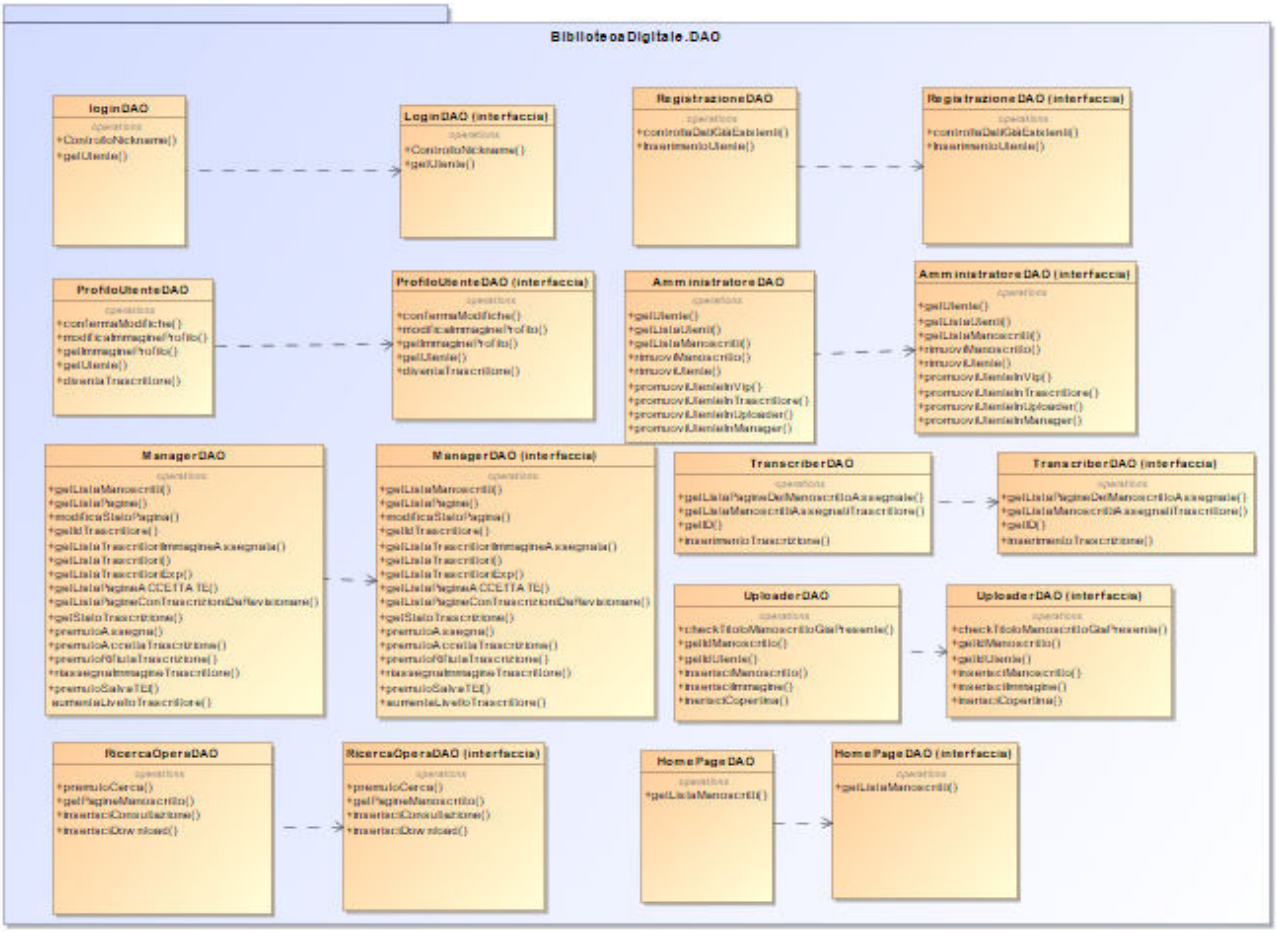
1-**Utente**: L’utente rappresenta la principale entità del sistema, in cui vengono descritte le principali funzionalità e caratteristiche che lo contraddistinguono.

2-**Manoscritto**: Il manoscritto rappresenta principalmente un testo di carattere divulgativo presente all’interno della piattaforma, rappresentato anche esso, cosi come per l’utente, attraverso una serie di informazioni e caratteristiche.

3-**Pagina**: Per pagina si intende, fondamentalmente, il corpo del Manoscritto, in cui figurano un’immagine di una pagina del Manoscritto, con la corrispondente trascrizione affianco.

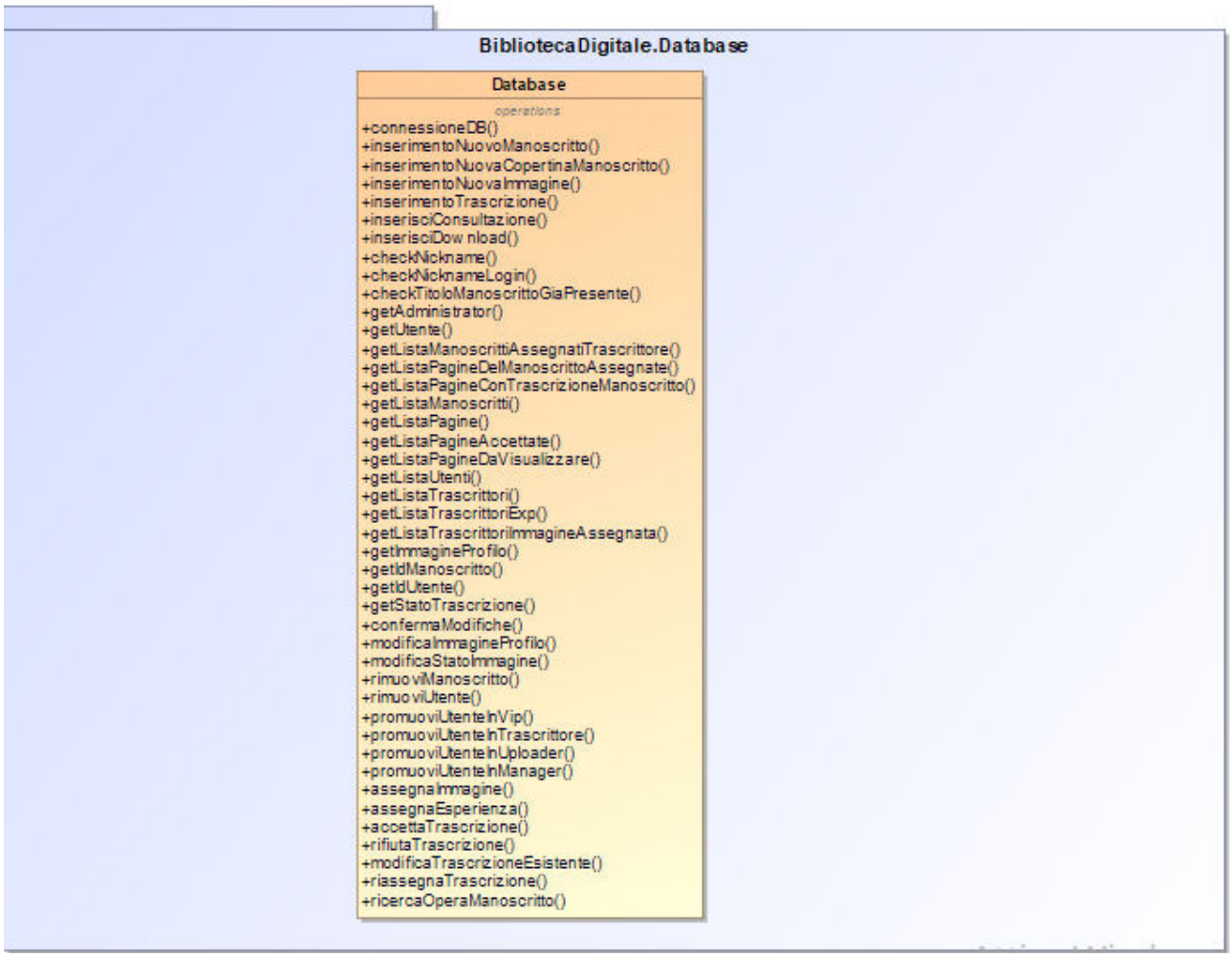
4-**Amministratore**: L’amministratore rappresenta un tipo di Utente, che tuttavia, svolge una sorta di ruolo di supervisore nei confronti dell’intera Piattaforma. Si ha un solo Amministratore per l’intera piattaforma e, come già accennato, egli ha il compito di gestire interamente gli utenti e i manoscritti.

Class Diagram – DAO:

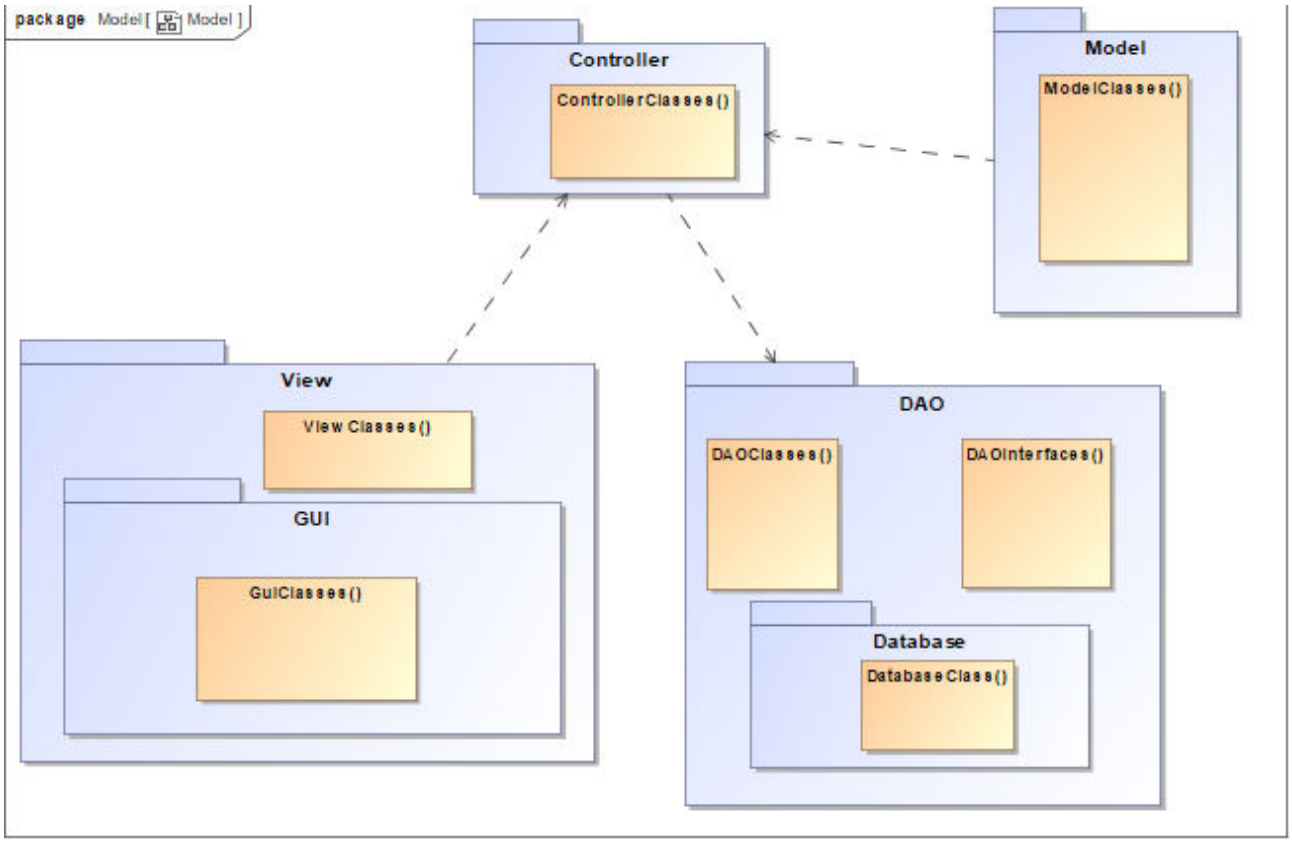


Il DAO rappresenta il principale package di comunicazione con il Database, in cui figurano, oltre alle classi, le rispettive interfacce. Un’interfaccia contiene i metodi (senza corpo) implementati dalle rispettive classi.

Class Diagram – Database:



Dopo una revisione del progetto, e sotto consiglio della Professoressa, si è deciso di modificare l’assettamento dei vari package. Si ottiene così il seguente Class Diagram completo:

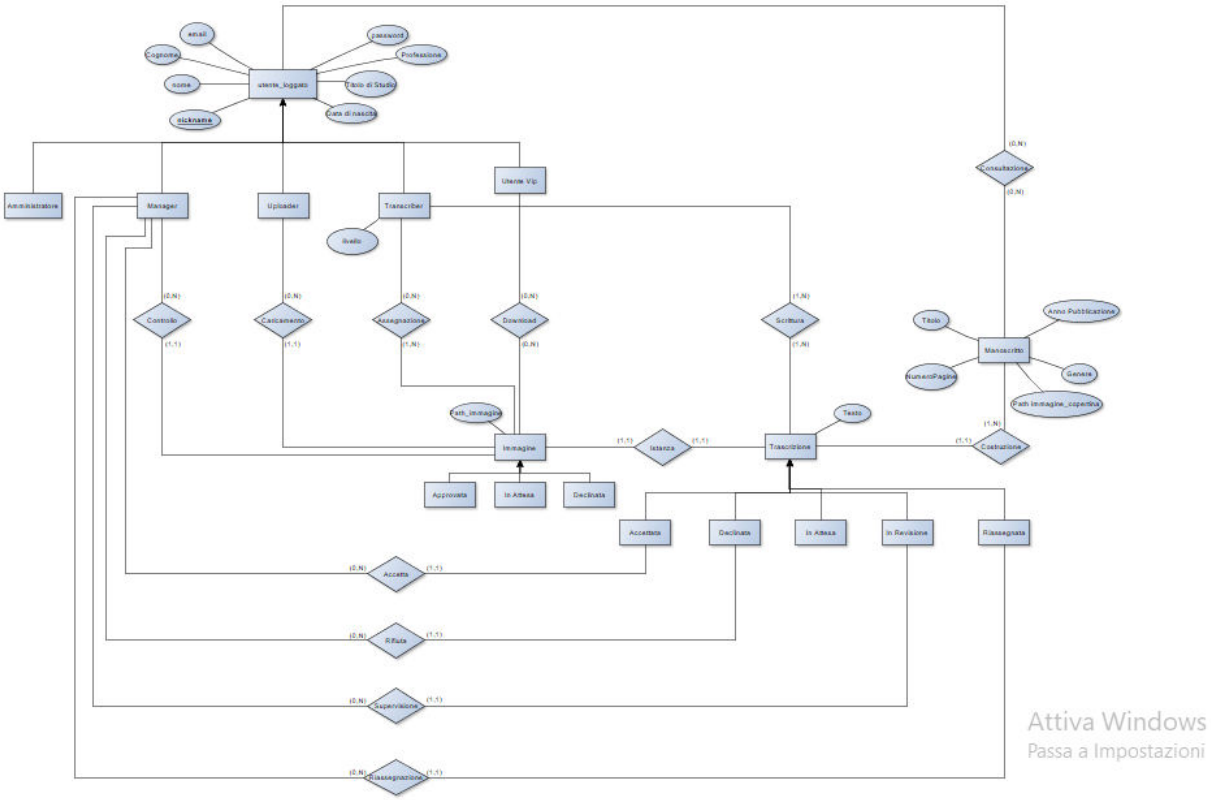


Database:

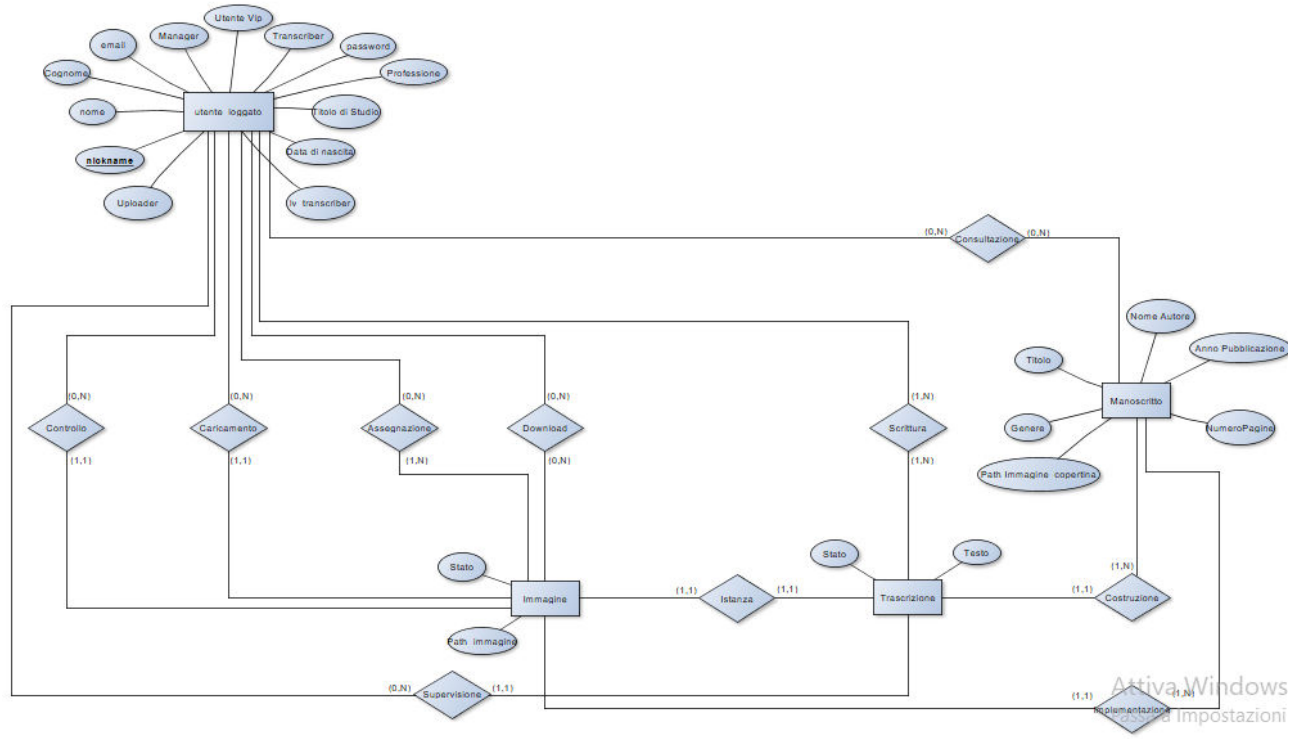
Il database risulta essere una delle componenti fondamentali della piattaforma. Al suo interno è possibile trovare tutte le varie tabelle utili a garantire persistenza dati, riguardanti soprattutto le entità più utilizzate e prese in considerazione come l’utente oppure il Manoscritto. E’ presente anche una tabella contenente tutti i Manoscritti opportunatamente inseriti, con le varie immagini e trascrizioni ad essi associati. La classe Database risulta anche essere la classe più utilizzata. Infatti, considerata la complessità del Database e delle operazioni svolte dai vari sottosistemi della piattaforma, il Database risulta essere aggiornato e interrogato numerose volte, per estrapolare informazioni e per modificare le stesse.

Modello Concettuale:

Seguendo una strategia di tipo “Top-Down” in cui si raggruppano i vari concetti partendo da uno schema “scheletro”, per poi farli “esplodere” in concetti più semplici, si è ottenuto il seguente modello concettuale:



Modello Logico:
Dopo le opportune modifiche ed eliminazioni di concetti presenti nel modello concettuale, come le generalizzazioni di entità, si ottiene alla fine il Modello Logico dei dati. Tale operazione si è resa necessaria in quanto, dal modello concettuale risultante al passo precedente, sono presenti concetti che non sono esprimibili nel Modello Concettuale che si vuole ottenere. Il modello logico risultante è il seguente:



Traduzione verso il Modello Relazionale:

Nel seguente Modello Relazionale sono riportate sia le tabelle sia le relazioni presenti all’interno del modello logico, le quali verranno rappresentate nel Database come normali tabelle.

Utente_loggato (ID, nome, cognome, data_nascita, email, nickname, password, utente_manager, utente_uploader, utente_transcriber,lv_transcriber, utente_vip, titolo_studio, Professione);

Amministratore (ID, nickname, nome, cognome, data_nascita, email, pass_word, professione);

Consultazione (ID_utente, ID_manoscritto);

Manoscritto (ID, titolo, numero_pagine, genere, nome_autore, anno_pubblicazione, path_immaginecopertina);

Immagine (ID, stato, path_immagine, **ID_manager**, **ID_uploader**, **ID_manoscritto**);

Download (ID, **ID_utente**, **ID_immagine**);

Trascrizione (ID, testo, stato, **ID_manager, ID_manoscritto, ID_immagine**);

Assegnazione (ID, **IDimmagine**, **IDtrascrittore**);

Scrittura (ID_trascrittore, ID_trascrizione);

Nello specifico:

-Gli attributi rappresentati sottolineati rappresentano le chiavi primarie delle varie tabelle;
-Gli attributi rappresentati in grassetto rappresentano le **chiavi esterne** delle varie tabelle;

Vincoli di integrità referenziale:

1-Dall’attributo ‘ID’ della tabella **Manoscritto** all’attributo ‘ID_manoscritto’ della tabella **Immagine**;

2-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_uploader’ della tabella **Immagine**;

3-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_manager’ della tabella **Immagine**;

3-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_utente’ della tabella **Download**;

4-Dall’attributo ‘ID’ della tabella **Immagine** all’attributo ‘ID_immagine’ della tabella **Download**;

5-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_manager’ della tabella **Trascrizione**;

6-Dall’attributo ‘ID’ della tabella **Manoscritto** all’attributo ‘ID_manoscritto’ della tabella **Trascrizione**;

7-Dall’attributo ‘ID’ della tabella **Immagine** all’attributo ‘IDimmagine’ della tabella **Trascrizione**;

8-Dall’attributo ‘ID’ della tabella **Immagine** all’attributo ‘IDimmagine’ della tabella **Assegnazione**;

9-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘IDtrascrittore’ della tabella **Assegnazione**;

10-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_utente’ della tabella **Consultazione**;

11-Dall’attributo ‘ID’ della tabella **Manoscritto** all’attributo ‘ID_manoscritto’ della tabella **Consultazione**;

12-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_trascrittore’ della tabella **Scrittura**;

13-Dall’attributo ‘ID’ della tabella **Trascrizione** all’attributo ‘ID_trascrizione’ della tabella **Scrittura**;

Creazione del Database e delle tabelle:

Software utilizzato: MySql WorkBench.

1-Creazione del Database:

```
/*Creazione Ddel Database relativo alla Biblioteca Digitale*/
drop database BibliotecaDigitaleOOSD;
create database BibliotecaDigitaleOOSD;
use BibliotecaDigitaleOOSD;
```

2-Tabella utente:

```
create table utente_loggato(
ID integer unsigned not null primary key auto_increment,
nickname varchar(30) not null,
immagine_profilo varchar(300) default null,
nome varchar(200) not null,
cognome varchar(200) not null,
data_nascita date,
email varchar(200) not null,
pass_word varchar(18) not null,
titolo_studio varchar(30) not null,
professione varchar(30) not null,
utente_vip boolean default false,
utente_manager boolean default false,
utente_transcriber boolean default false,
lv_transcriber integer unsigned default null,
utente_uploader boolean default false,
constraint email_unica unique (email),
constraint nickname_unico unique (nickname)
);
```

3-Tabella Amministratore:

```
create table Amministratore(
ID integer unsigned not null primary key auto_increment,
nicknameAmministratore varchar(30) not null default 'Administrator',
nome varchar(200) not null,
cognome varchar(200) not null ,
data_nascita date not null ,
email varchar(200) not null ,
pass_word varchar(18) not null default 'Administrator',
professione varchar(30) not null
);
```

4-Tabella Manoscritto:

```
create table Manoscritto(
ID integer unsigned not null primary key auto_increment,
titolo varchar(200) not null,
immagine_copertina varchar(500) default null,
anno_pubblicazione varchar(4) not null,
numero_pagine integer unsigned not null,
genere varchar(200) not null,
nome_autore varchar(200) not null
);
```

5-Tabella Consultazione:

```
create table consultazione(
ID_utente integer unsigned not null,
ID_manoscritto integer unsigned not null,
primary key (ID_utente,ID_manoscritto),
constraint lettura_utente foreign key (ID_utente) references utente_loggato(ID),
constraint lettura_pagina foreign key (ID_manoscritto) references Manoscritto(ID)
);
```

6-Tabella Immagine:

```
create table immagine(
ID integer unsigned not null primary key auto_increment,
path_immagine varchar(500) not null,
ID_uploader integer unsigned not null,
ID_manoscritto integer unsigned not null,
stato enum('accettata','declinata','In Attesa') not null default 'In Attesa',
ID_manager integer unsigned default null,
constraint manager_immagine foreign key(ID_manager) references utente_loggato(ID),
constraint uploader_immagine foreign key(ID_uploader) references utente_loggato(ID),
constraint manoscritto_immagine foreign key(ID_manoscritto) references manoscritto(ID) on delete cascade on update cascad
);
```

7-Tabella Download:

```
create table download(
ID integer unsigned not null primary key auto_increment,
ID_utente integer unsigned not null,
ID_immagine integer unsigned not null,
constraint download_utente foreign key (ID_utente) references utente_loggato(ID),
constraint download_immagine foreign key (ID_immagine) references immagine(ID)
);
```

8-Tabella Trascrizione:

```
create table trascrizione(
ID integer unsigned not null primary key auto_increment,
testo blob,
stato enum('accettata','declinata','riassegnata','in revisione') not null,
ID_manager integer unsigned default null,
ID_manoscritto integer unsigned not null,
ID_immagine integer unsigned not null,
constraint trascrizione_manager foreign key(ID_manager) references utente_loggato(ID),
constraint trascrizione_manoscritto foreign key(ID_manoscritto) references Manoscritto(ID),
constraint trascrizione_immagine foreign key (ID_immagine) references immagine(ID)
);
```

9-Tabella Assegnazione:

```
create table assegnazione(
ID integer unsigned not null primary key auto_increment,
IDimmagine integer unsigned not null,
IDtrascrittore integer unsigned not null,
constraint assegnazione_immagine foreign key (IDimmagine) references immagine(ID),
constraint assegnazioneimmagine_trascrittore foreign key (IDtrascrittore) references utente_loggato(ID)
);
```

10-Tabella Scrittura:

```
create table scrittura(
ID_trascrittore integer unsigned not null,
ID_trascrizione integer unsigned not null,
primary key (ID_trascrittore, ID_trascrizione),
constraint scrittura_trascrittore foreign key (ID_trascrittore) references utente_loggato(ID),
constraint scrittura_trascrizione foreign key (ID_trascrizione) references Trascrizione(ID)
);
```

Progetto Object-Oriented Software Design

Progetto “Biblioteca Digitale”

A.A. 2017/2018

Cicerone Loreto – De Cesaris Marco – Caruso Simone

1)Specifiche:

Il progetto è un’estensione della traccia fornita nell’a.a. 2015/16 e si propone di realizzare una biblioteca digitale di testi e studi che contribuiscono alla formazione della cultura all’interno dell’Università degli Studi dell’Aquila. Una biblioteca digitale è uno spazio in cui mettere insieme collezioni, servizi e persone a supporto dell'intero ciclo di vita di creazione, uso, preservazione di dati, informazione e conoscenza. Lo scopo di questo progetto è di consentire la consultazione dei manoscritti che devono essere digitalizzati e che costituiscono un patrimonio bibliografico antico per un totale di 60.000 carte (ms.sec.XV-XIX) contenenti memorie storiche della città dell’Aquila. Il sistema si divide nei seguenti sottosistemi:

1. Viewer
2. Uploader
3. Transcriber
4. Manager
5. Administrator

I sottosistemi sono indipendenti tra loro ma comunicano al fine di realizzare i loro scopi.

In particolare, vengono descritti come segue.

Viewer:

Tale parte del sistema consente la consultazione delle opere digitali a utenti registrati. Consente la ricerca nel catalogo per metadati (descritti in seguito) oppure all’interno del testo della trascrizione. Le opere possono essere suddivisi in categorie. Appena si sceglie un’opera, verrà visualizzata con una schermata che avrà sulla destra il testo della trascrizione (se disponibile) e sulla sinistra l’immagine della pagina dell’opera che si sta visualizzando, sarà possibile sfogliare le pagine tramite un paginatore. Alcuni utenti con particolari privilegipossono effettuare il download dell’opera. L’utente può fare richiesta tramite un modulo per essere collaboratore del sistema (trascrittore). Gli utenti possono accedere al loro profilo personale dove saranno visualizzati i dati inseriti nella registrazione, tra cui: titolo di studio, professione, indirizzo, email, etc.

Uploader:

Ogni opera è formata da più immagini (scansioni), ognuna delle quali rappresenta una pagina del manoscritto. Per ogni opera vengono caricati dei metadati (titolo, anno, …). Al fine di rendere agevole il caricamento delle immagini e il successivo controllo, per ogni opera si vuole fornire la visualizzare sia tutte le pagine in miniatura e di una pagina per volta da scorrere con un paginatore. La digitalizzazione viene controllata da supervisori all’acquisizione per assicurarne la correttezza (ad esempio, in accordo con standard richiesti) e la qualità.

Transcriber:

Ogni opera acquisita deve essere trasformata in un testo digitale, ciò avviene attraverso operazioni di trascrizioni in formato TEI (Text Encoding Initiative). Le trascrizioni sono digitate manualmente attraverso un text editor TEI integrato. Quindi ogni immagine (pagina) avrà il corrispondente testo associato. Più trascrittori possono lavorare sulla stessa pagina, è necessario sincronizzare le modifiche. Le trascrizioni sono oggetto di revisione da parte di revisori alle trascrizioni.

Manager:

Questo sottosistema gestisce le assegnazioni, ovvero consente di assegnare parte di un’opera (1 o più immagini) a 1 o più trascrittori. Inoltre, consente di revisionare le trascrizioni concluse, di effettuare correzioni e validazione. E’ possibile anche riassegnare delle pagine ai trascrittori. Consente la pubblicazione delle trascrizioni e delle opere (solo immagini). Gestisce i livelli dei trascrittori (ogni trascrittore ha un livello 1-5 in base alla sua esperienza). Consente la supervisione dell’acquisizione immagini.

Administrator:

Gestione in back-end di tutto il sistema: anagrafica utenti, opere, etc.

1.1) Analisi dei requisiti:

Individuazione dei concetti fondamentali:

Il progetto è un’estensione della traccia fornita nell’a.a. 2015/16 e si propone di realizzare una biblioteca digitale di testi e studi che contribuiscono alla formazione della cultura all’interno dell’Università degli Studi dell’Aquila. Una biblioteca digitale è uno spazio in cui mettere insieme collezioni, servizi e persone a supporto dell'intero ciclo di vita di creazione, uso, preservazione di dati, informazione e conoscenza. Lo scopo di questo progetto è di consentire la consultazione dei manoscritti che devono essere digitalizzati e che costituiscono un patrimonio bibliografico antico per un totale di 60.000 carte (ms.sec.XV-XIX) contenenti memorie storiche della città dell’Aquila. Il sistema si divide nei seguenti sottosistemi:

1. Viewer
2. Uploader
3. Transcriber
4. Manager
5. Administrator

I sottosistemi sono indipendenti tra loro ma comunicano al fine di realizzare i loro scopi.

In particolare, vengono descritti come segue.

Viewer:

Tale parte del sistema consente la consultazione delle opere digitali a utenti registrati. Consente la ricerca nel catalogo per metadati (descritti in seguito) oppure all’interno del testo della trascrizione. Le opere possono essere suddivisi in categorie. Appena si sceglie un’opera, verrà visualizzata con una schermata che avrà sulla destra il testo della trascrizione (se disponibile) e sulla sinistra l’immagine della pagina dell’opera

che si sta visualizzando, sarà possibile sfogliare le pagine tramite un paginatore. Alcuni utenti con particolari privilegi possono effettuare il download dell’opera. L’utente può fare richiesta tramite un modulo per essere collaboratore del sistema (trascrittore). Gli utenti possono accedere al loro profilo personale dove saranno visualizzati i dati inseriti nella registrazione, tra cui: titolo di studio, professione, indirizzo, email, etc.

Uploader:

Ogni opera è formata da più immagini (scansioni), ognuna delle quali rappresenta una pagina del manoscritto. Per ogni opera vengono caricati dei metadati (titolo, anno, …). Al fine di rendere agevole il caricamento delle immagini e il successivo controllo, per ogni opera si vuole fornire la visualizzazione sia tutte le pagine in miniatura e di una pagina per volta da scorrere con un paginatore. La digitalizzazione viene controllata da supervisori all’acquisizione per assicurarne la correttezza (ad esempio, in accordo con standard richiesti) e la qualità.

Transcriber:

Ogni opera acquisita deve essere trasformata in un testo digitale, ciò avviene attraverso operazioni di trascrizioni in formato TEI (Text Encoding Initiative). Le trascrizioni sono digitate manualmente attraverso un text editor TEI integrato. Quindi ogni immagine (pagina) avrà il corrispondente testo associato. Più trascrittori possono lavorare sulla stessa pagina, è necessario sincronizzare le modifiche. Le trascrizioni sono oggetto di revisione da parte di revisori alle trascrizioni.

Manager:

Questo sottosistema gestisce le assegnazioni, ovvero consente di assegnare parte di un’opera (1 o più immagini) a 1 o più trascrittori. Inoltre, consente di revisionare le trascrizioni concluse, di effettuare correzioni e validazione. E’ possibile anche riassegnare delle pagine ai trascrittori. Consente la pubblicazione delle trascrizioni e delle opere (solo immagini). Gestisce i livelli dei trascrittori (ogni trascrittore ha un livello 1-5 in base alla sua esperienza). Consente la supervisione dell’acquisizione immagini.

Administrator:

Gestione in back-end di tutto il sistema: anagrafica utenti, opere, etc.

Glossario dei termini:

Termine	Descrizione
Viewer	Utente correttamente registrato nel sistema e con la quale può liberamente interagire entro i suoi limiti.
Opere	Manoscritti che possono essere visionati dagli utenti registrati nel sistema.
Ricerca	Il sistema deve offrire l’opportunità all’utente di effettuare ricerche di Manoscritti per metadati.
Trascrizione	Trascrizione di un’immagine eseguita attraverso un TEI in

	caratteri digitali.
Immagine	Immagine di una certa pagina del Manoscritto.
Download	Un utente con particolari privilegi può effettuare lo scaricamento di una certa immagine di una pagina del Manoscritto.
Profilo Personale	L’utente deve avere la possibilità di poter visionare i propri dati, opportunatamente registrati all’interno del suo profilo personale, accessibile dalla piattaforma.
Uploader	Utente che ha il compito di caricare immagini di pagine di un certo Manoscritto.
Metadati	Campi che identificano un certo Manoscritto all’interno del sistema.
Transcriber	Utente che ha la possibilità di poter trascrivere in caratteri digitali una certa immagine caricata all’interno della piattaforma utilizzano un TEI.
T.E.I.	Text-Editor opportunatamente creato con il quale è possibile trascrivere le immagini caricate.
Manager	Uno dei più importanti sottosistemi della piattaforma. Egli ha il dovere di revisionare il lavoro svolto dall’uploader e dal transcriber.
Assegnazioni	Il manager, tra i suoi compiti, deve assegnare delle immagini ai transcriber, i quali ne effettueranno la trascrizione.
Revisione	Attività svolta dal manager, la quale consiste nell’accettare o rifiutare immagini o trascrizioni inserite.
Ri/assegnare	Attività svolta dal manager, la quale consiste nell’assegnare o nel riassegnare immagini

	caricate nel sistema a più transcriber, che quindi possono trascrivere la stessa immagine. Il lavoro sarà valutato direttamente dal Manager.
Administrator	Amministratore del sistema, il quale si occupa della gestione dell’intera piattaforma.

1.2) Documento dei requisiti:

Il sistema si pone come obiettivo quello di offrire un servizio che consenta la consultazione, da parte degli utenti, di antichi manoscritti, il tutto come una biblioteca virtuale. L’utente deve necessariamente effettuare una registrazione nella biblioteca virtuale, mettendo tutti i vari campi consoni ad una corretta registrazione, ed in seguito può effettuare un login. Tra gli utenti che si iscrivono possono esserci utenti con particolari privilegi, che, in particolare, hanno la possibilità di effettuare il download di un’immagine del manoscritto che desiderano. I dati degli utenti possono essere consultati dagli stessi quando si vuole, grazie ad una sezione “profilo utente” fornita direttamente dal sistema. L’utente, inoltre, riempiendo un apposito modulo, può avere la possibilità di diventare un transcriber. Il sistema prevede la visualizzazione del manoscritto nel seguente modo: Una volta scelto il manoscritto da voler visualizzare saranno presenti a sinistra e destra, rispettivamente, l’immagine di una pagina del manoscritto e la rispettiva trascrizione al suo fianco, quest’ultima effettuata dal transcriber. Le immagini sono caricate dall’uploader, il quale ha un ruolo diverso rispetto al transcriber, infatti esso è incaricato solo del caricamento delle immagini. Il sistema consente di visualizzare quest’ultime sia in miniatura, sia per intero, con un opportuno paginatore che consente di andare avanti e indietro nello scorrimento delle immagini delle pagine. L’uploader ha a disposizione 4 tipi di metadati da caricare per ogni opera: Autore, titolo, anno, genere. Una volta eseguito questo passo, le varie immagini saranno visionate dal manager, anch’esso un altro sottosistema, che ne esaminerà la correttezza. Una volta che le immagini caricate dall’uploader saranno accettare dal manager, sarà compito del transcriber elaborare le immagini traducendole in caratteri digitali.
Il sistema prevede che ogni opera avrà una o più pagine al suo interno, che possono essere visualizzate come mosaico oppure per intero. Tali pagine saranno così formate: A sinistra c’è l’immagine selezionata, mentre a destra c’è l’editor di testo dove è possibile trascrivere il testo dell’immagine in formato TEI. Ogni transcriber avrà un grado di esperienza ad esso associato, riportato in una scala da 1 a 5, e visualizzabile all’interno della sezione profilo fornita dal sistema. Le varie pagine vengono assegnate al transcriber grazie al manager, il quale può decidere anche di assegnare la stessa pagina a

transcriber diversi. Una volta effettuata la trascrizione della pagina, sarà compito del manager verificare la correttezza e la validazione della trascrizione. Spetterà quindi al manager verificare l’esattezza del lavoro svolto dall’uploader e dal transcriber. Il sistema prevede inoltre che la sezione utente, con tutti i vari campi che lo identificano, e la sezione delle opere, anch’esse verificate da campi come Autore, Titolo, Anno e genere, siano gestite da un amministratore della piattaforma. Il sottosistema più importante della piattaforma è però l’Amministratore, il quale ha il compito di gestire l’intera piattaforma.

I requisiti funzionali del progetto sono i seguenti, indicati da una scala di priorità da 1 a 5 (1: priorità minima, 5: priorità massima):

1) **Registrazione e Login**: L’utente deve registrarsi per poter usufruire dei vari servizi (Priorità 5).

2) **Univocità**: Nel sistema non possono comparire utenti con nickname uguali, evitando, quindi, ambiguità nel sistema stesso. (Priorità 5).

3) **Utente VIP**: Nel sistema possono esserci utenti con particolari privilegi che possono scaricare un’immagine di un’opera. (Priorità 1).

4) **Profilo Utente**: L’utente deve avere la possibilità di poter modificare i suoi campi e generalità (memorizzati in un opportuno Database). (Priorità 3).

5) **Diventa Transcriber**: Il sistema deve dare la possibilità agli utenti di poter diventare transcriber, quindi nel sistema ce ne deve essere almeno uno che si occupi della trascrizione. L’utente può diventare transcriber solo dopo aver compilato l’apposito modulo fornitogli, oppure promosso grazie al lavoro dell’amministratore (Priorità 5).

6) **Sezione visualizzazione opera**: Il sistema deve dare la possibilità di poter visualizzare le immagini dell’opera per intero, con relativa trascrizione affianco (Priorità 5).

7) **Declinazione immagine**: Il sistema deve fornire all’utente immagini di buona e non di scarsa qualità caricate dall’uploader, aspetto che può rendere difficile la trascrizione. Ne consegue che il manager può declinare il caricamento di immagini di bassa qualità (Priorità 4).

8) **Univocità dell’immagine**: Il sistema deve dare ad una immagine del manoscritto un numero che la identifichi univocamente all’interno dell’opera, compito svolto, in particolare, dall’uploader. (Priorità 4).

9) **Ricerca Opera**: Il sistema deve dare la possibilità all’utente di poter eseguire una ricerca dell’opera che a lui interessa (Priorità 5).

I requisiti non funzionali sono riportati di seguito:

1) **Usability**: Usabilità del sistema, ovvero la facilità per l’utente di imparare ad usare il sistema rapidamente, fornendo, quindi, un’interfaccia user-friendly.

2) **Reliability**: Affidabilità del sistema, ovvero la reazione a casi limite, garantendo le funzionalità messe a disposizione senza errori. Ad esempio il sistema deve poter bloccare la registrazione di un utente nel caso in cui i 2 abbiano nickname uguali.

3) **Performance**: Velocità di esecuzione del sistema, ovvero il lavoro che il sistema riesce ad eseguire in una data unità di tempo.

1.3) Attori del sistema:

Utente Loggato: L’utente è il primo attore che è messo in risalto dal sistema. Esso, come già specificato, per interagire con la piattaforma, deve necessariamente effettuare una registrazione, ed in seguito un login. Esso ha, inoltre, la possibilità di diventare un **Utente Vip,** e quest’ultimo, in particolare, ha la possibilità di poter scaricare le immagini dell’opera che più gli interessano. L’utente può consultare il suo profilo utente quando desidera, e può, in caso, modificare i suoi campi (generalità e altro) quando desidera. Inoltre, compilando un apposito modulo che apparirà sempre nella sua sezione **profilo utente**, ha la possibilità di diventare **Transcriber.** Il transcriber ha la possibilità di poter trascrivere le immagini di singole pagine dell’opera che gli vengono assegnate dal **manager** (descritto in seguito).

Transcriber: Il transcriber, come già descritto, è semplicemente un utente già iscritto nel sistema, il quale ha la possibilità di trascrivere opere. Il manager può assegnare l’immagine da caricare nel sistema a più transcriber, che quindi possono aver modificato più volte la trascrizione. Sarà compito sempre del **manager** scegliere quale opera sia più consona. Il transcriber inoltre ha associato un grado di esperienza che può variare da 1 a 5.

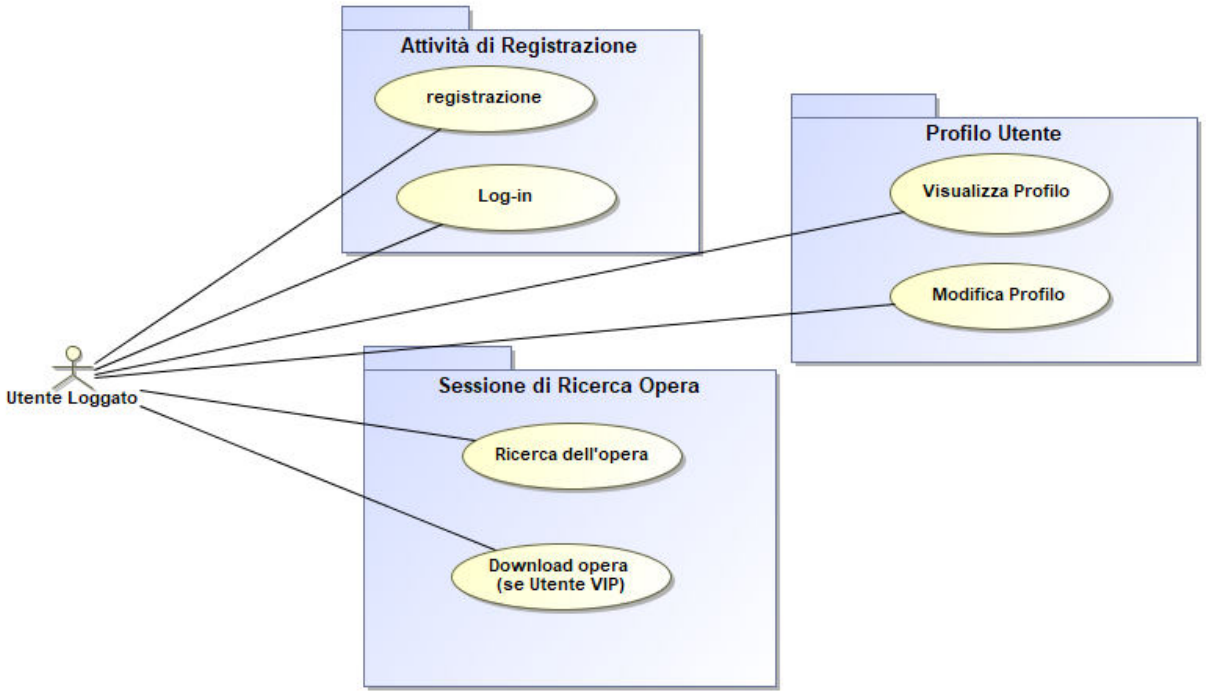
Manager: Il manager è uno degli attori più importanti del sistema. Cosi come un normale utente, egli ha la possibilità di modificare e visualizzare il suo profilo. Egli, inoltre, ha il dovere di dover gestire il grado di esperienza dei transcriber, di visionare la trascrizione delle opere effettuando nel caso anche correzioni, ma anche di accettare nel sistema immagini delle varie opere che si dovranno trascrivere (possibilmente immagini di almeno buona qualità). Questo aspetto del caricamento delle immagini, è svolto da un altro attore del sistema, **l’uploader.**

Uploader: Anche l’uploader può essere considerato come un utente della piattaforma. Il suo compito è quello di caricare immagini nel sistema. Ogni immagine caricata deve avere un numero che la identifichi univocamente all’interno del sistema. Inoltre, ogni immagine deve essere facilmente distinguibile all’interno dell’opera. A tal proposito si può pensare di identificare ogni immagine attraverso un numero che la identifichi univocamente all’interno dell’opera stessa.

Amministratore: L’amministratore è senza dubbio l’attore più importante all’interno del sistema, infatti ha la completa gestione dell’intera piattaforma. Egli ha la possibilità di eliminare utenti all’interno del sistema, interagendo con un database, e di promuovere gli stessi ad **Utenti Vip**, secondo opportuni privilegi che questi hanno. Ha anche la possibilità di modificare ed eliminare opere, anch’esse memorizzate all’intero del database.

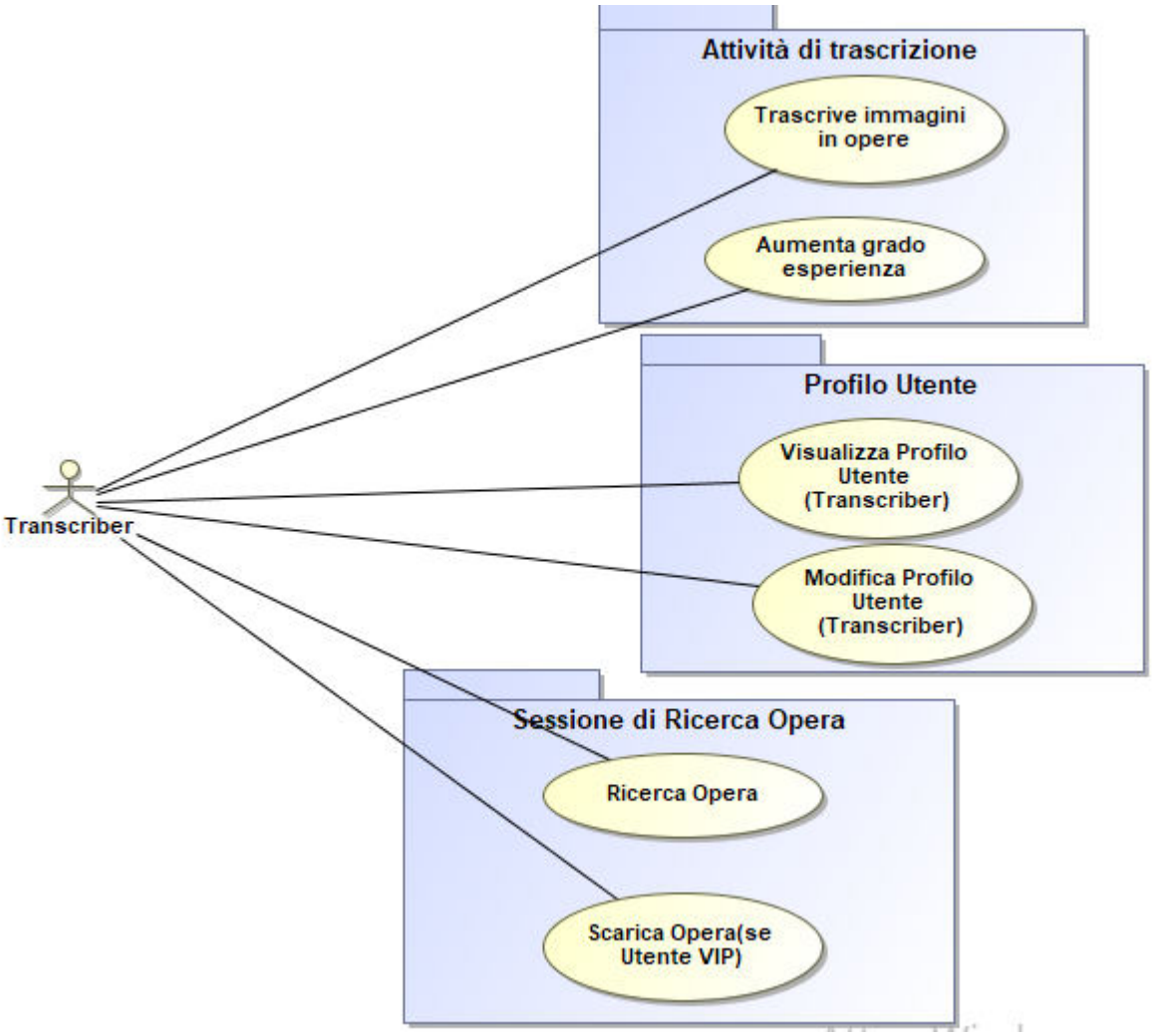
1.4) Use-Case:

Use-case per l’utente:



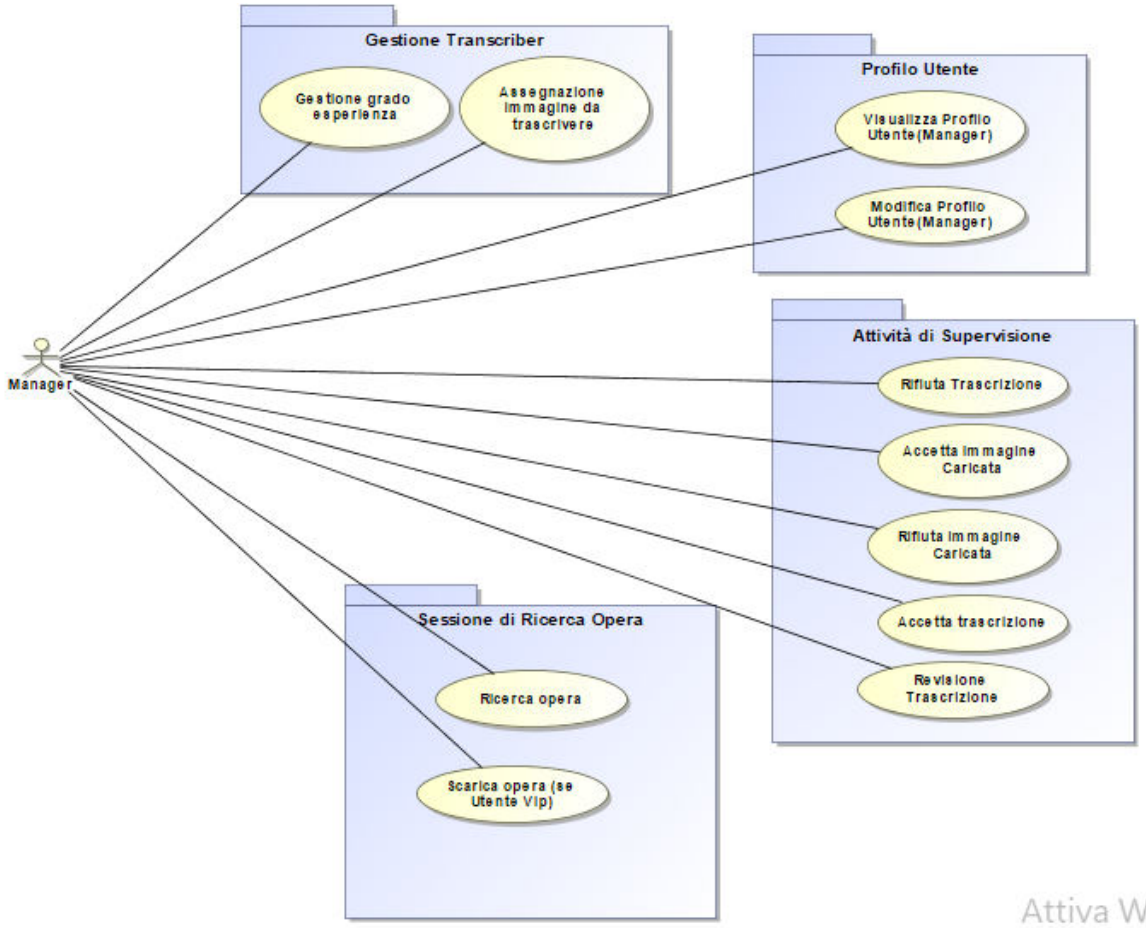
L’utente, dopo essersi iscritto nel sistema, può visualizzare il suo profilo, diventare transcriber, e può modificare il suo profilo (riportato nel package “**Profilo Utente**”). Inoltre può effettuare una ricerca dell’opera, e se è un **Utente Vip**, può scaricare l’opera (riportato nel package “**Sessione di ricerca opera**”).

Use case per il transcriber:



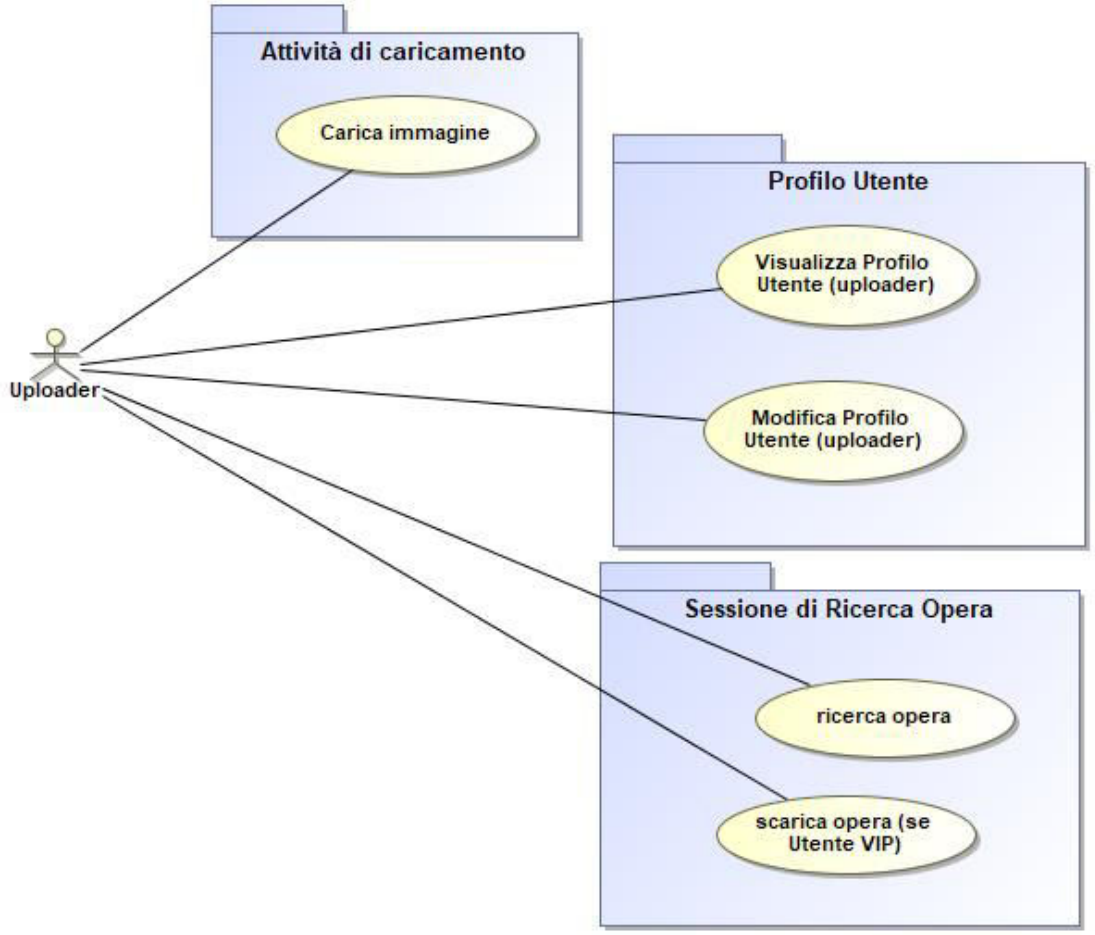
Un utente registrato che è anche transcriber dopo aver compilato l’apposito modulo, può svolgere le stesse azioni di un normale **utente loggato**, inoltre svolge le azioni di un transcriber, ossia trascrivere opere ed aumentare il suo grado di esperienza (quest’ultimo aspetto è gestito, in particolare, dal manager).

Use case per il manager:



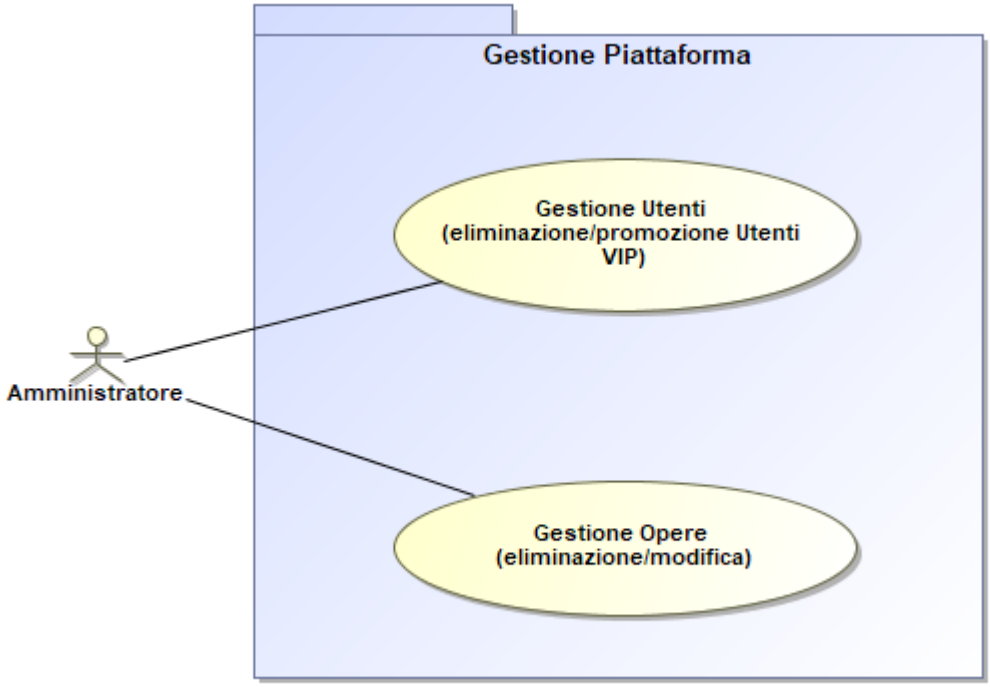
Un manager può svolgere le stesse funzionalità di un utente loggato. Inoltre egli svolge attività si supervisione come quelle riportate in figura. Infine egli assegna il lavoro ai transcriber, aumentandone, in caso, il grado di esperienza.

Use case per l’uploader:



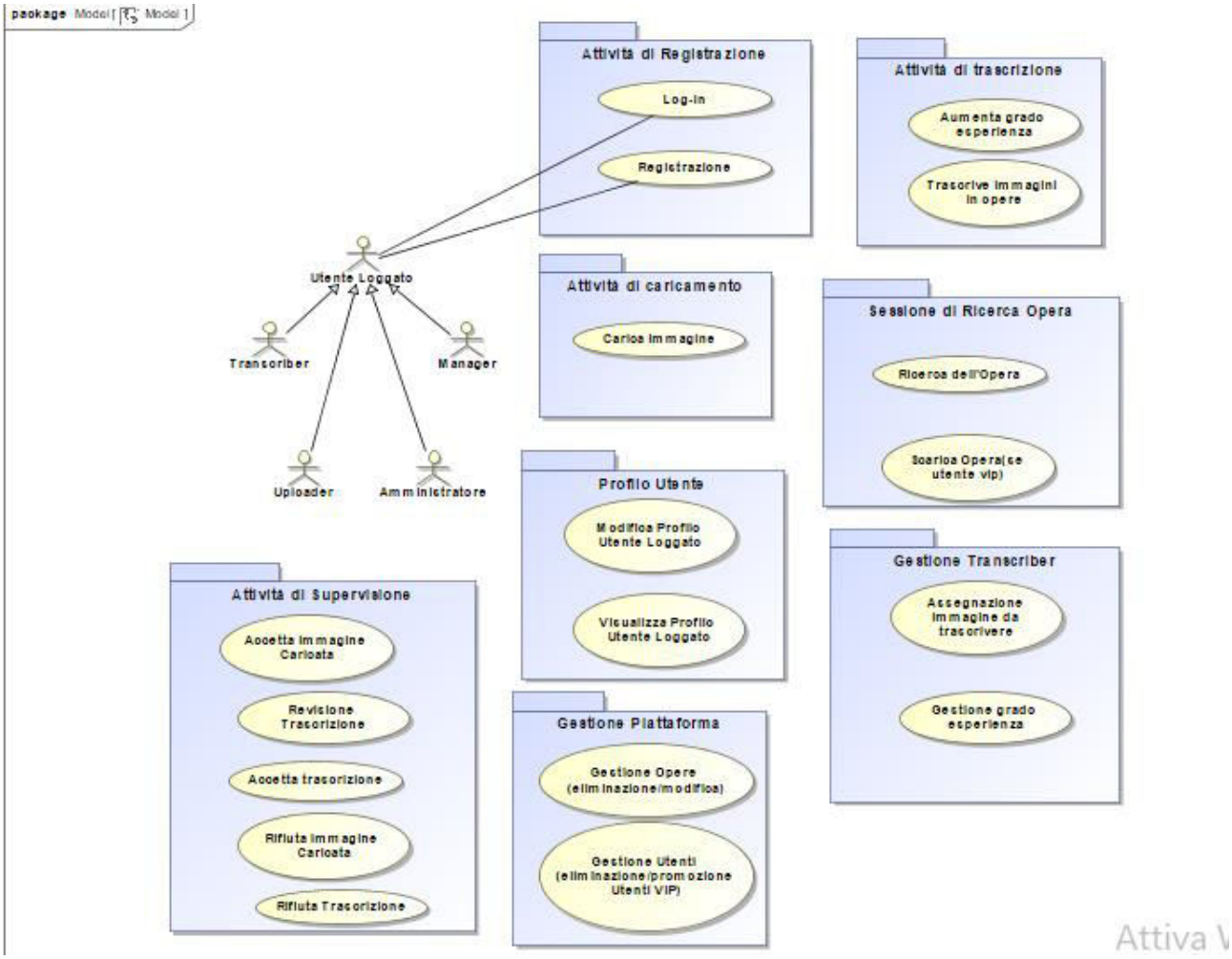
Un uploader svolge le stesse azioni di utente. In più egli ha il compito di caricare le immagini delle singole pagine di un’opera.

Use case per l’amministratore:



L’amministratore si può considerare come l’attore più importante del sistema. Egli svolge attività di gestione dell’intera piattaforma e, in particolare, si occupa dell’eliminazione e della modifica di opere. Inoltre ha la possibilità di gestire gli utenti e di promuoverli ad utenti vip, i quali hanno possibilità di scaricare immagini dell’opera che desiderano.

Use-case completo:



Lo use-case completo comprende tutti gli use-case mostrati in precedenza.
N.B: Si noti che le linee di associazione tra attore e use-case non sono state messe di proposito per non “sporcare” il disegno e renderlo di difficile comprensione. La descrizione dello use-case completo sarà espressa negli Scenari.

1.5) Scenari:

Attività di Registrazione: La registrazione nella piattaforma è svolta da tutti gli attori presenti nel sistema, inserendo opportuni dati consoni ad una corretta registrazione (anagrafica e altro) e scegliendo, inoltre, uno username ed una password. Si noti che, come descritto nei requisiti funzionali, si è deciso di scegliere nel sistema solo utenti che abbiano nickname che li identifichino univocamente, ciò significa che non possono esserci utenti con nickname uguali.

Profilo Utente: Tutti gli utenti avranno un loro profilo utente, il quale consentirà loro di visualizzare le proprie generalità e i propri ruoli all’interno della piattaforma. La sezione profilo utente permette agli stessi anche di modificare il proprio profilo.

Attività di Trascrizione: L’attività di Trascrizione, da come si può evincere, è svolta dal transcriber, il quale si occupa di trascrivere immagini in opere.

Attività di caricamento: L’attività di caricamento è svolta dall’uploader, il quale ha il solo compito di caricare l’immagine di una pagina dell’opera di appartenenza.

Sessione di Ricerca Opera: Questa attività può essere svolta da tutti gli utenti del sistema, e consiste nel fornire un servizio di ricerca dell’opera che si desidera (se presente nel sistema).

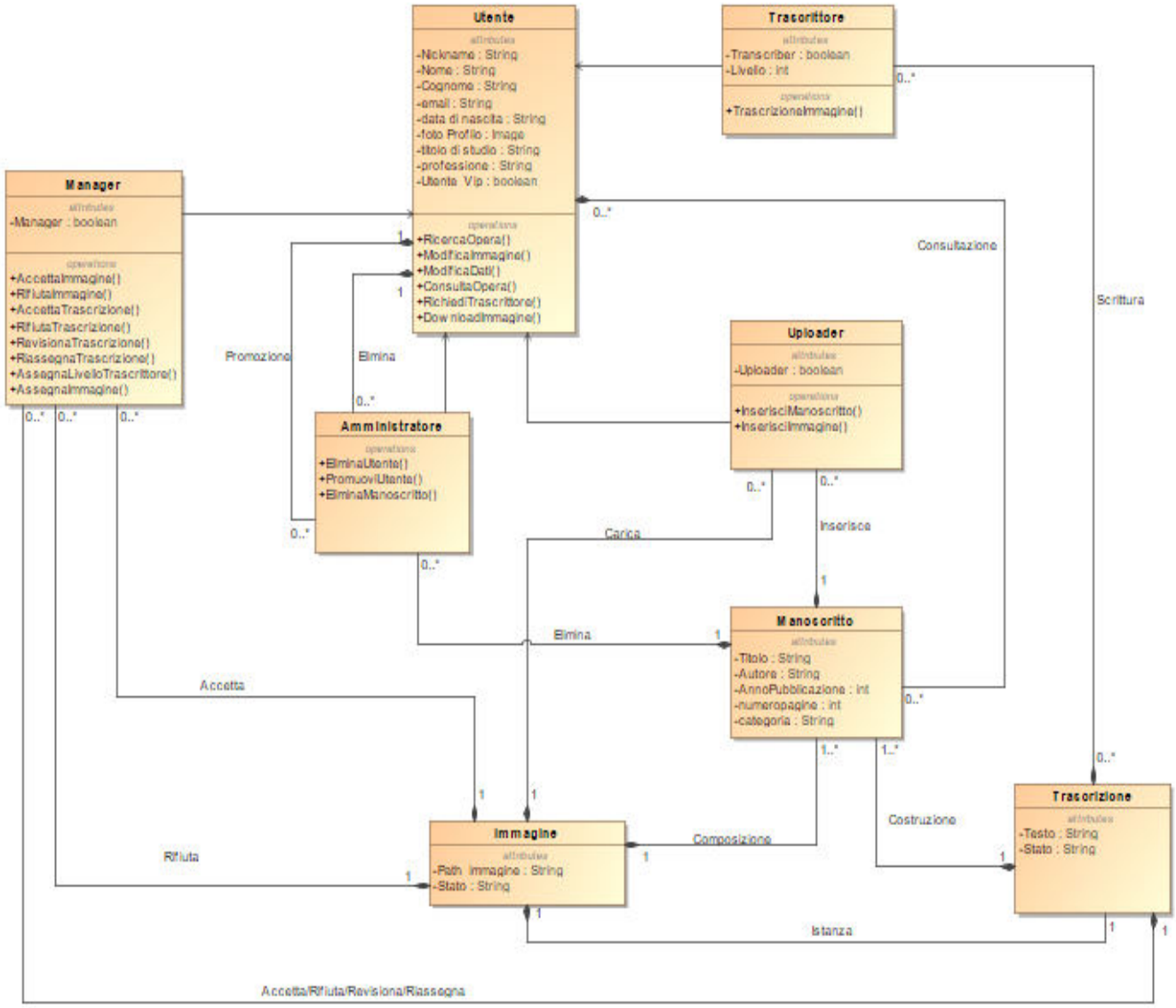
Gestione Transcriber: Questa attività è svolta dal manager, e consiste nell’assegnare un’opera da trascrivere ad un transcriber e gestirne il suo grado di esperienza.

Attività di Supervisione: L’attività di supervisione è svolta sempre dal manager, e consiste nella supervisione dei vari lavori che lo stesso manager ha assegnato ad uploader e transcriber, come ad esempio l’accettazione, il rifiuto o la modifica di una trascrizione o il declinamento di un’immagine caricata dall’uploader, che, ad esempio, non è di sufficiente qualità.

Gestione Piattaforma: La gestione della piattaforma è svolta dall’amministratore del sistema, che come già accennato, può essere considerato come l’attore più importante all’interno della piattaforma stessa. Infatti egli ha pieno controllo del sistema, infatti è in grado di modificare/cancellare il profilo di un utente o di un’ opera.

1.6) **Modello di Dominio**:

Per modello di dominio si intende un insieme di classi utili a rappresentare e finalizzare i problemi che si possono presentare nella costruzione di un sistema, come ad esempio nel nostro caso:



Nel seguente modello vengono riportate tutte le classi scelte che si identificano all’interno del sistema, riportando, per ognuna di esse, i vari attributi e i vari metodi. Nel modello presentato sono presenti opportune relazioni tra le varie classi, in cui in più, sono riportate le varie associazioni. Sono presenti anche le opportune generalizzazioni tra le varie classi, come ad esempio i vari tipi di utente che ci possono essere all’interno del sistema, generalizzati, appunto, dalla classe utente. Nel seguente dominio sono anche state messe relazioni di composizione, come ad esempio tra la classe manoscritto e la classe pagina. Una relazione di composizione, nell’esempio, identifica l’esistenza obbligatoria di un manoscritto affinchè esistano una o più pagine, altresì, se non vi è alcun manoscritto, non possono esistere pagine ad esso associate.

1.8) Individuazione di classi entity, boundary e controller:

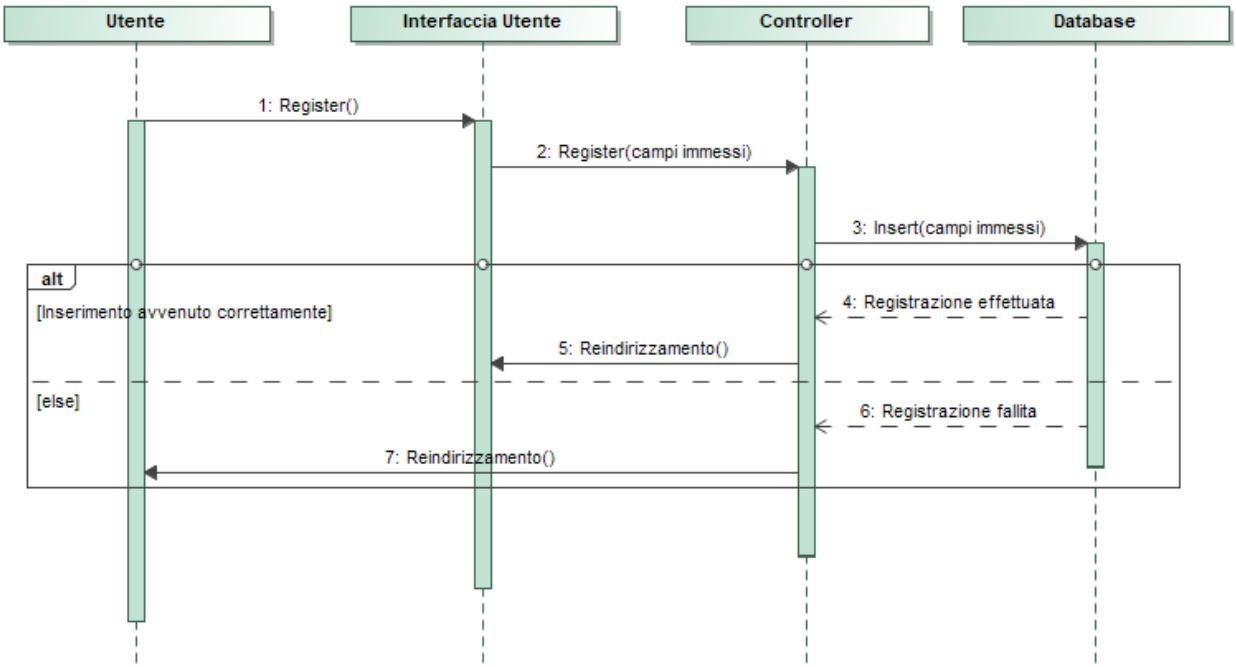
Le classi entity del sistema possono essere considerate le stesse presenti all’interno del modello di dominio. Sarà presente una classe **Utente**, la quale conterrà i vari attributi e operazioni che caratterizzano l’utente stesso. Le classi **Utente vip, trascrittore, uploader, amministratore e manager** sono sottoclassi della classe principale Utente, quindi tutte le sottoclassi citate erediteranno gli stessi attributi e operazioni della classe padre utente. In aggiunta, le sottoclassi avranno un attributo boolean, il quale assumerà valori true o false in base al ruolo che gli utenti registrati svolgono all’interno della piattaforma. Le altre classi entity scelte sono:

Manoscritto: Questa classe rappresenta, appunto, un manoscritto all’interno della piattaforma. Essa avrà i vari attributi che identificano lo stesso manoscritto, come l’autore del manoscritto, il titolo del manoscritto, l’anno di pubblicazione, numero di pagine al suo interno, ecc... Inoltre esso avrà un campo id al suo interno, il quale lo identificherà univocamente all’interno del sistema.

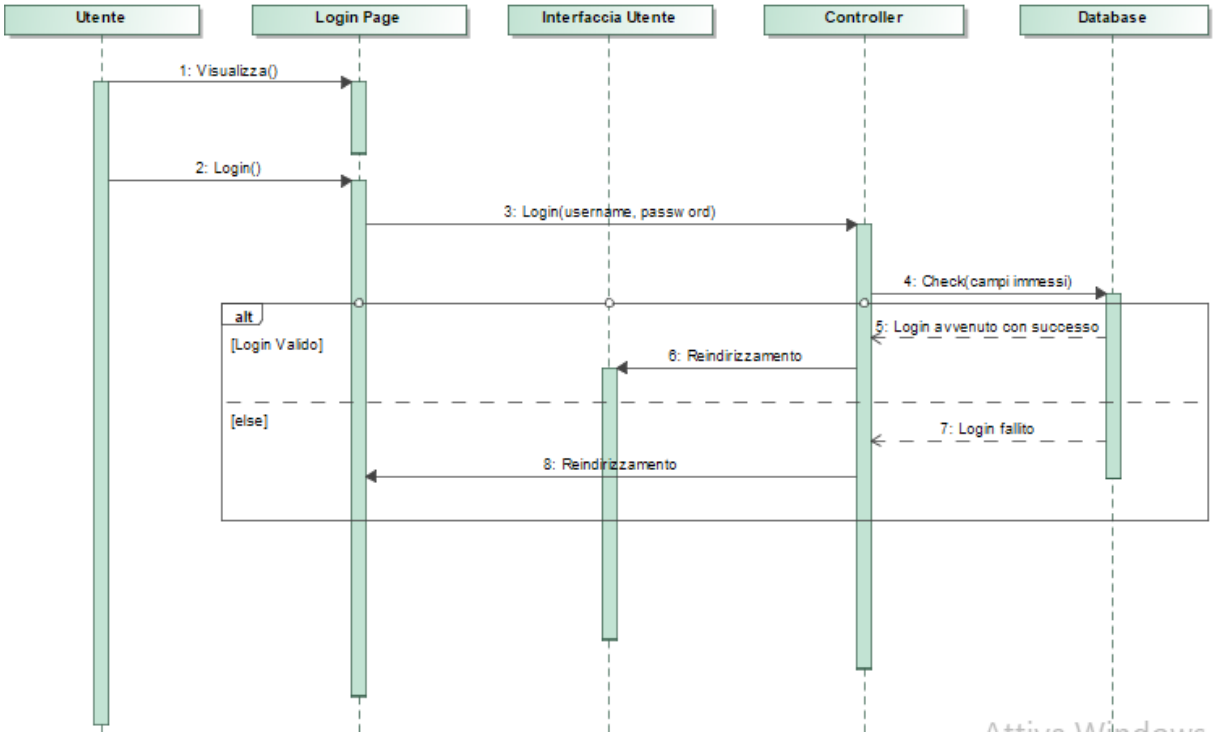
Trascrizione: Questa classe identifica il testo della trascrizione relativa ad una immagine, è quindi opportuno che essa abbia un campo testo.

Immagine: La classe Immagine identifica una certa immagine di una pagina di un’opera all’interno della piattaforma. Ad una immagine possono essere associati un attributo qualità, in base al quale il manager può declinare il caricamento dell’immagine stessa, ed un attributo che identifica una variabile image, la quale servirà a contenere l’immagine vera e propria.

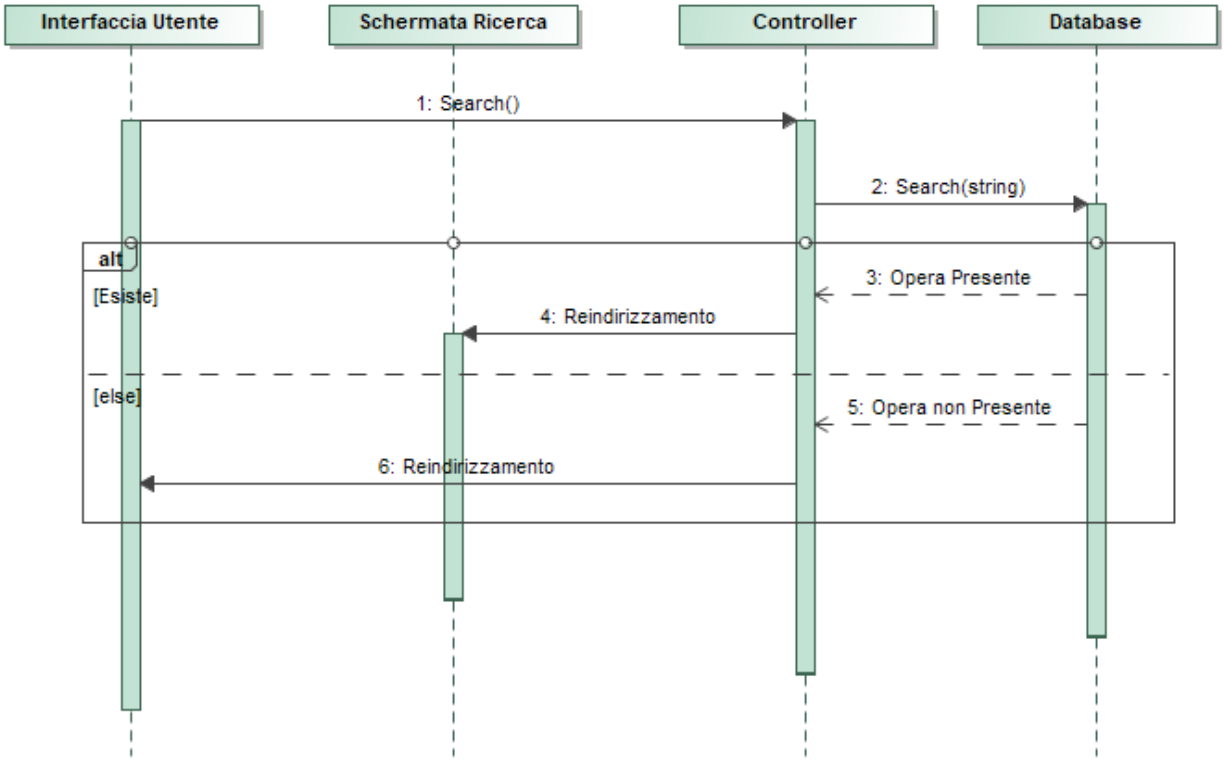
Le classi citate saranno opportunatamente gestite e mantenute in un Database. Le classi boundary identificheranno le varie interfacce che faranno da tramite tra il sistema e l’utente, come ad esempio, finestre, popup di avviso e menù. Il tutto sarà all’interno di un unico package GUI (Graphical User Interface). Le classi Controller si occuperanno del corretto controllo del flusso di informazioni tra le classi boundary e le classi entity nel sistema. Quindi ad ogni classe boundary del package GUI verrà assegnata una classe Controller del package Controller.
Una rappresentazione del funzionamento di tali classi può essere rappresentato nei **Sequence Diagram** illustrati di seguito.



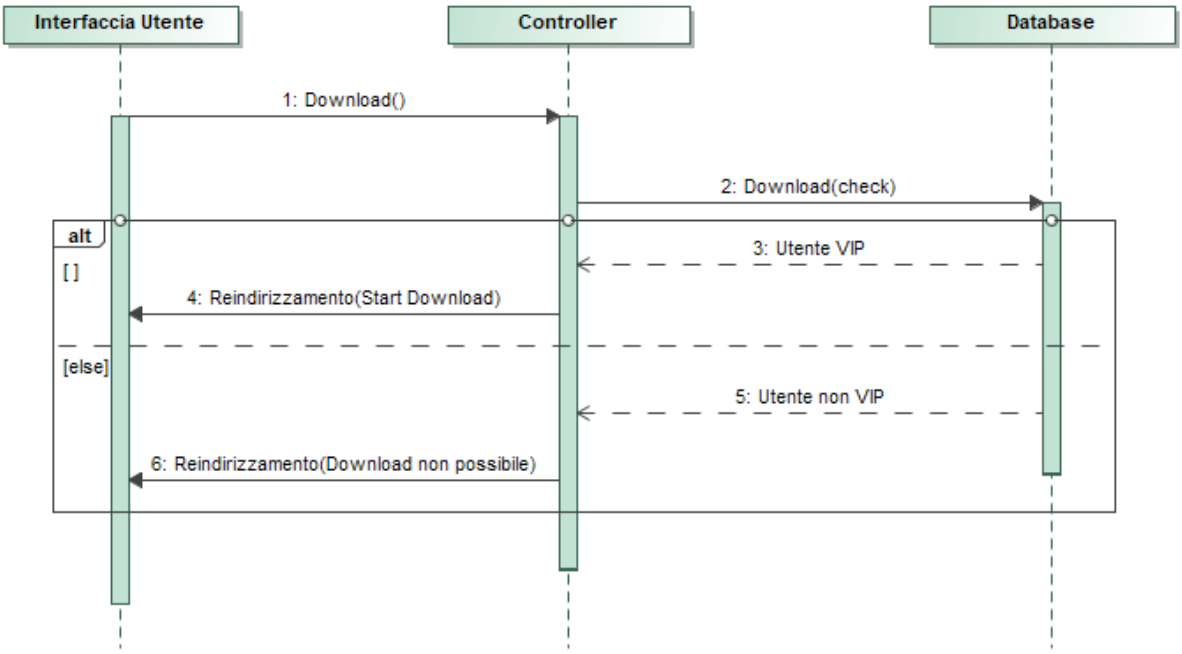
1-Registrazione Utente.



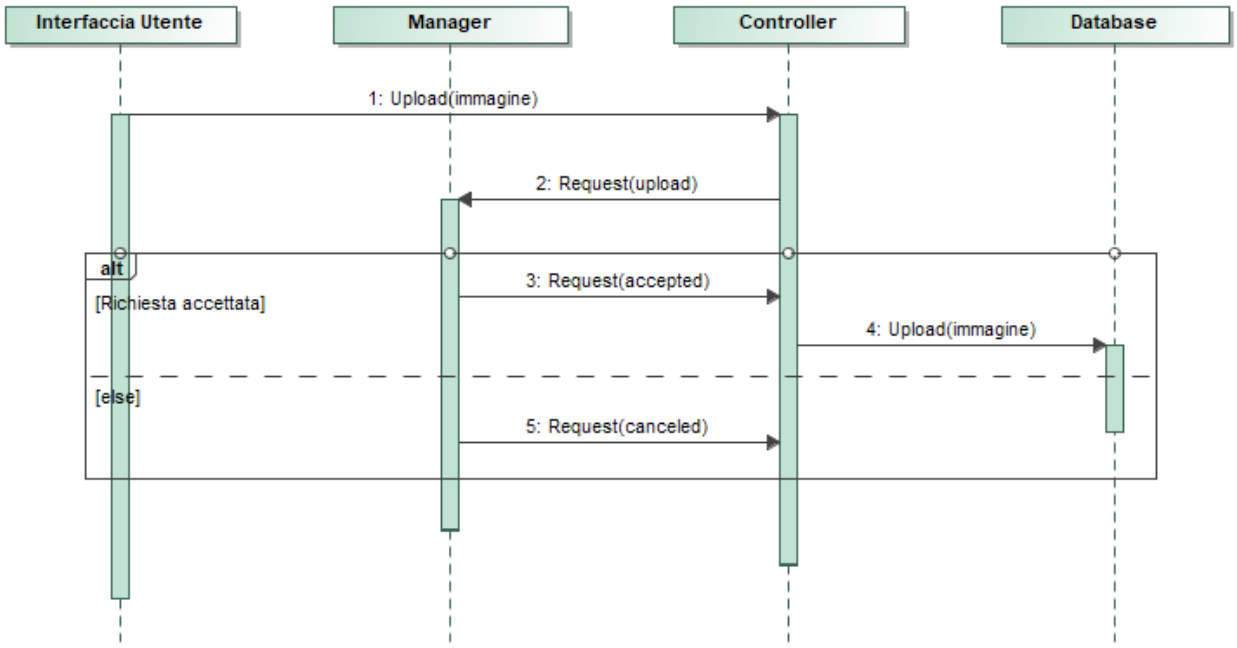
2-Login Utente



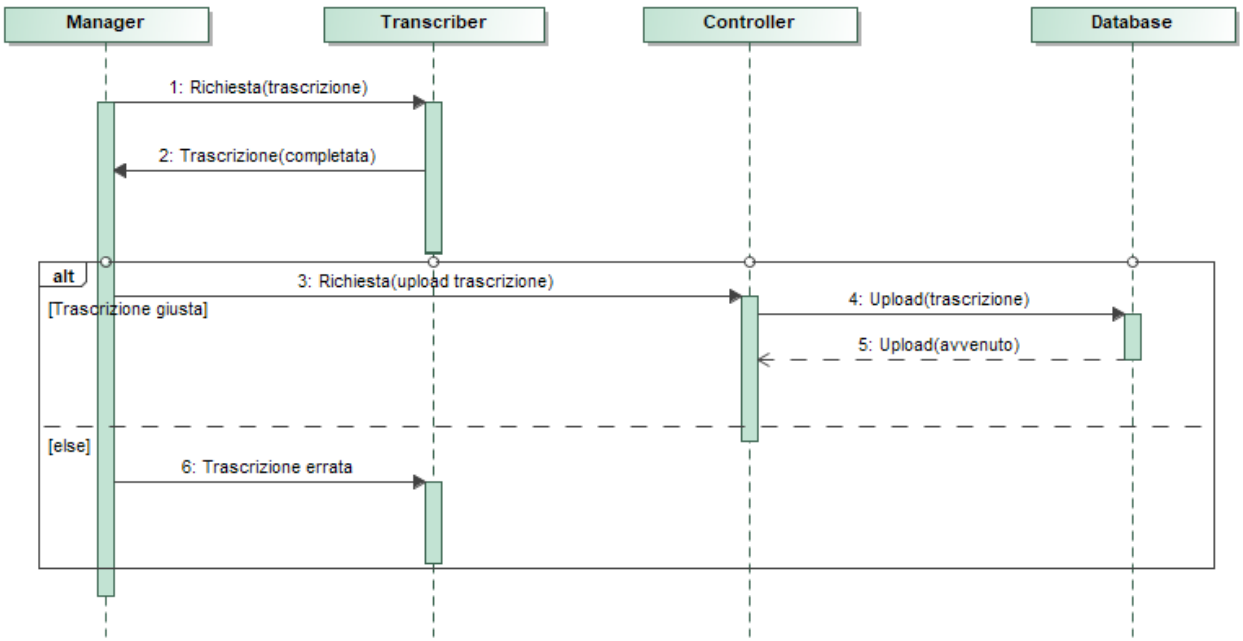
3-Ricerca Opera



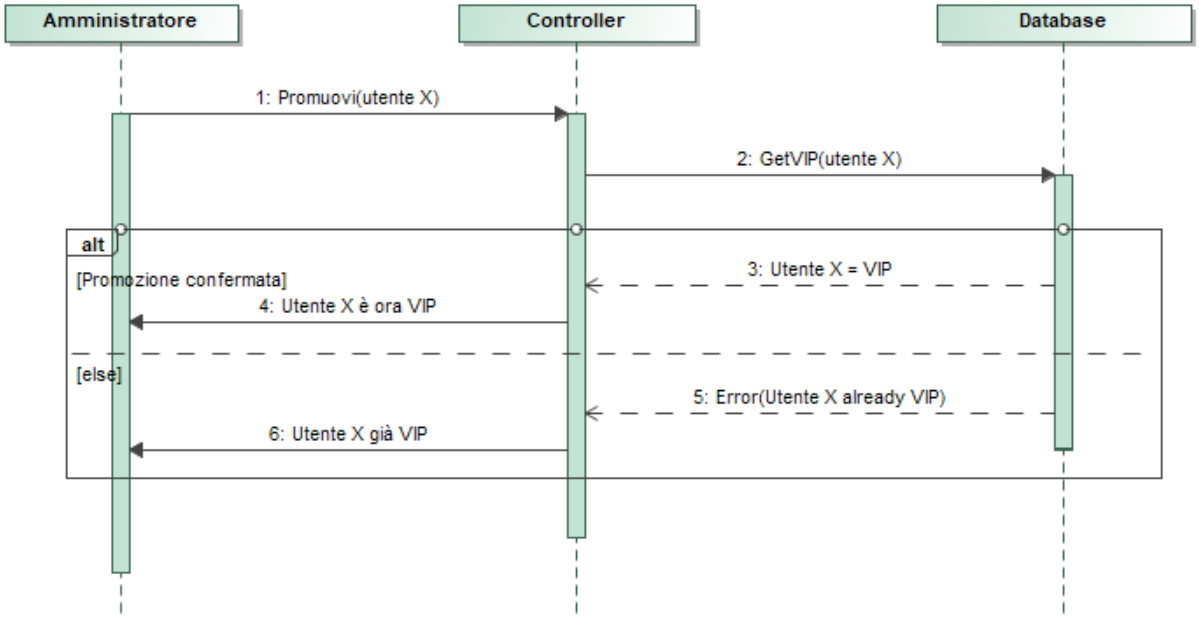
4-Download Opera.



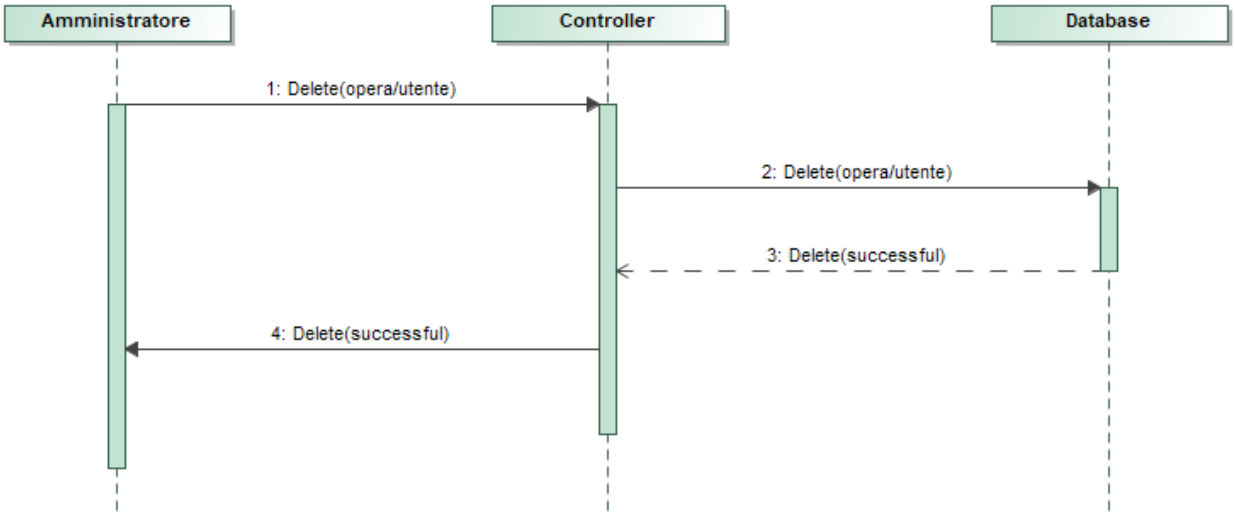
5-Upload immagine



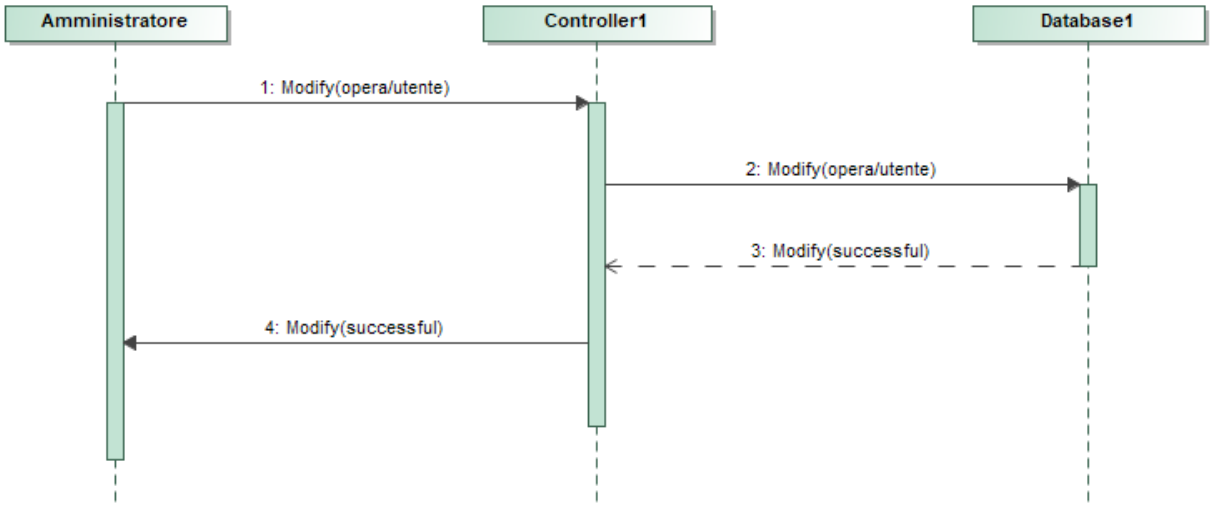
6- Caricamento Trascrizione.



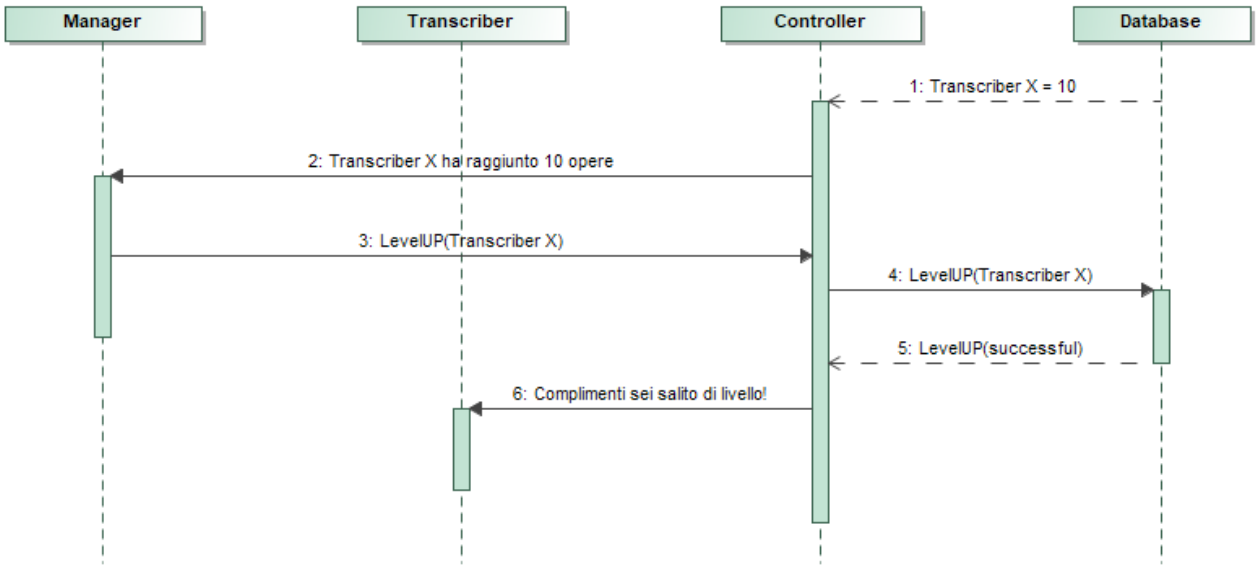
7- Promozione Utente VIP



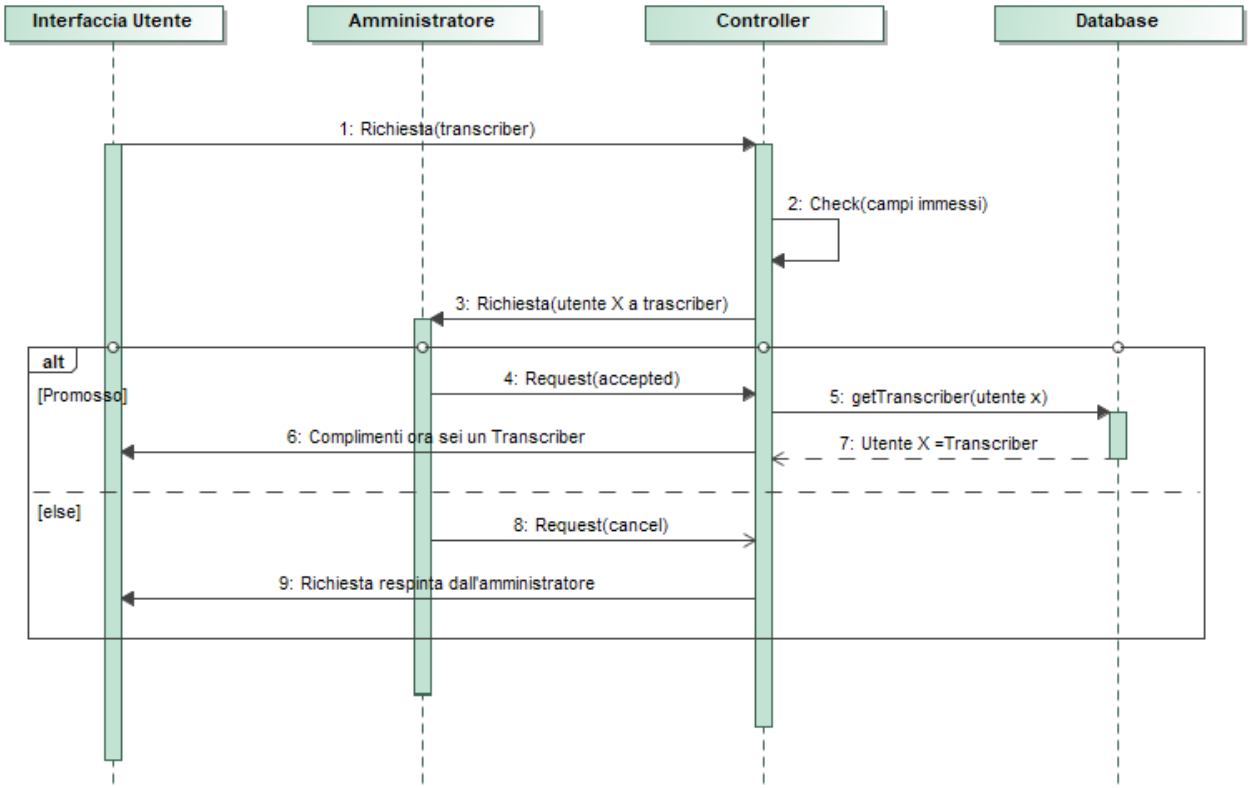
8- Eliminazione opera/utente.



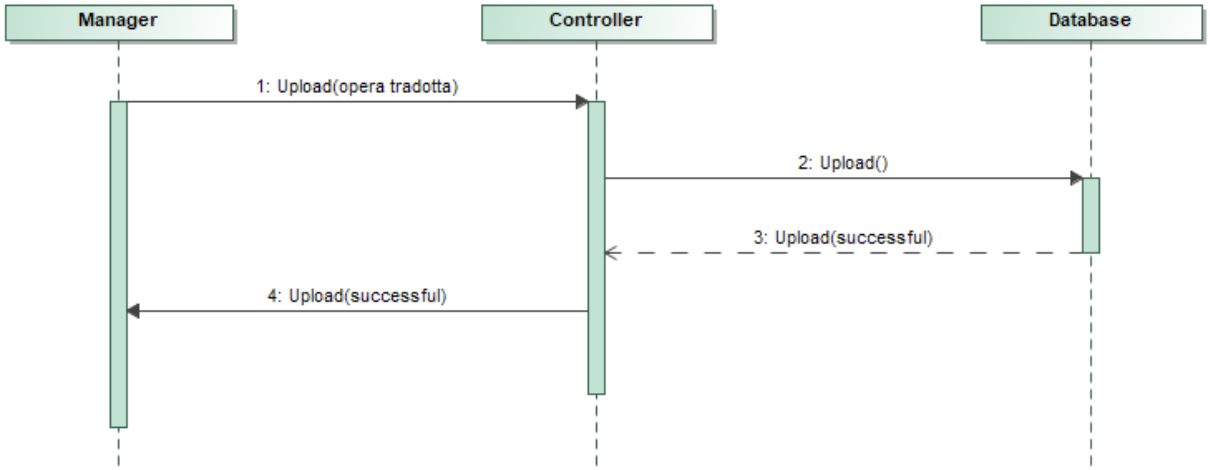
9- Modifica opera/utente.



10- Avanzamento livello esperienza Transcriber.



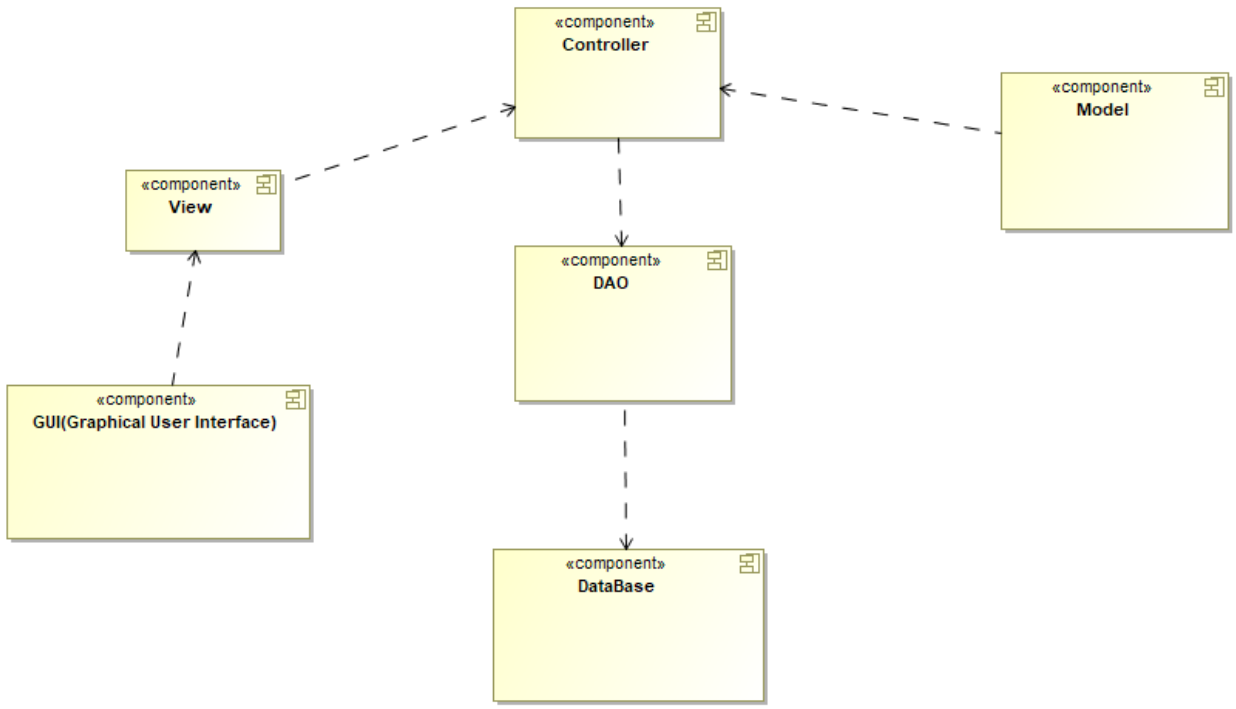
11-Diventa Transcriber



12-Pubblicazione Opera.

2.1) System Design

Per la descrizione dell’architettura software del sistema si è scelto di usare un **Component Diagram**:



2.2) Descrizione dell’architettura:

La componente **GUI (Graphical User Interface)**, come già descritto, identifica le varie interfacce che faranno da tramite tra il sistema e l’utente. La componente **View** sarà identificata da un altro package, e più approfonditamente la view verrà implementata in ogni entità della GUI e verrà chiamata in campo per un primo controllo sull’input utente, come previsto dal pattern **MVC (Model View Controller)**. Quindi, ad esempio, la classe profiloUtente, implementerà utente e controllerProfiloUtente. In particolare, la componente View sarà utilizzata dal controller per un corretto utilizzo dei dati e una corretta gestione di essi da parte dello stesso utente attraverso il costrutto ***implements***. Il tutto sarà gestito in modo da essere di facile apprendimento e user-friendly (vedi requisito non funzionale **Usability**). La già citata componente **Controller** si occuperà del corretto flusso di dati immessi dall’utente, effettuando, nel caso, re-indirizzamenti e gestione delle funzionalità nel sistema, come già dimostrato nei vari **Sequence Diagram.** La componente **Model** avrà al suo interno tutti i vari oggetti utilizzati dalle altre classi, ossia quelle già descritte e specificate nel **Domain Model**. Questo avviene mediante funzioni che hanno il compito di interagire con gli oggetti selezionati. Per esempio, quando si seleziona da una lista un oggetto e si avvia, per esempio, una certa funzione “cancella oggetto”, attraverso tale funzione si va a interagire con la classe in questione eliminandola dal contesto. Questo flusso di azioni è gestito dalla componente associata alla componente Model, il Controller. L’altra componente che incontriamo è il **Data Access Object (DAO)**. Il

DAO è la frontiera più esterna che separa il codice Java dall’effettiva e fisica componente del sistema **Database**, e tramite l’implementazione di alcune librerie specifiche di Java sarà possibile la corretta comunicazione tra Java e MySQL.

2.3) Descrizione delle scelte e strategie adottate:

Il problema principale da esporre è la gestione e l’ITER-procedurale dei dati immessi dall’utente, questo perché si vuole garantire un accesso al Database pulito. Per fare ciò abbiamo identificato 4 sezioni distinte:

1- Il **Model** rappresenta tutti i vari attori del sistema, per esempio utenteVip() extends utente() ecc... fanno parte delle entità su cui verranno fatte delle modifiche, si interagirà, e in generale saranno partecipe di varie azioni.

2- La **GUI** sarà l’unica interfaccia con cui l’utente si relazionerà direttamente. Attraverso l’implementazione delle classi GUI con classi di tipo Controller avverrà la comunicazione dei dati tra le prime con le seconde. In più tale classe per raccogliere o settare i vari dati dovrà necessariamente implementare la classe Model con cui interagisce ed ereditarne i relativi metodi di settaggio e di return dei vari attributi.

3- La **View** avrà il compito di controllare il primo input fatto dall’utente, come già descritto sopra. Tale controllo sarà possibile grazie all’implementazione di tale classe nelle singole classi GUI. Per fare ciò la view avrà varie funzioni che verranno richiamate nel momento del settaggio degli attributi, se l’esito del controllo dell’input è positivo la view invierà tale input al Controller.

4- Il **Controller** fa da prima linea per input errati nella forma (es. caratteri non ammessi/errati). In tale evenienza si rida il controllo alla GUI attraverso la view che ne evidenzia a video l’errore riscontrato. Caso in cui il primo check dell’input viene accettato si comunicano tali attributi alla componente DAO per accertare, per esempio, la già presente esistenza di un eventuale nickname all’interno del Database. Quindi, effettivamente il controller ha un ruolo di prima scrematura e controllo dei dati.

5- Il **DAO** è un’interfaccia di Java che permette un corretto e più semplice utilizzo del Database fisico, qualunque esso sia. Il DAO riceve dati dal Controller, e su questi dati effettua query passandole alla gestione del Database che si sceglie.

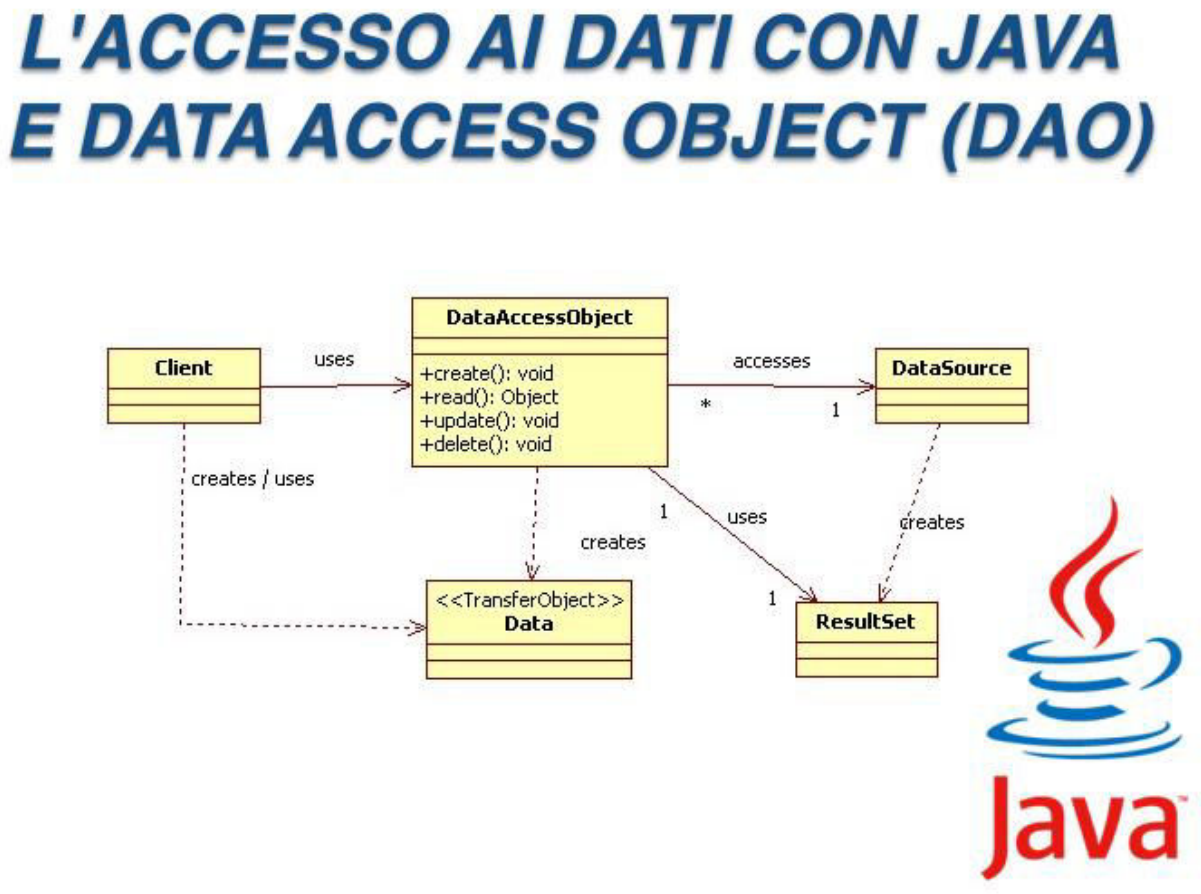
6- Il **Database** da noi utilizzato sarà MySQL. Esso garantirà persistenza, affidabilità e integrità dei dati relativi alle varie entità nel sistema.

2.4) Design Patterns:

Data-Access-Object:

Si è scelto di usare come design pattern il DAO (**D**ata **A**ccess **O**bject) per gestire, in particolare, la persistenza dei dati nel Database. Il pattern si basa su:

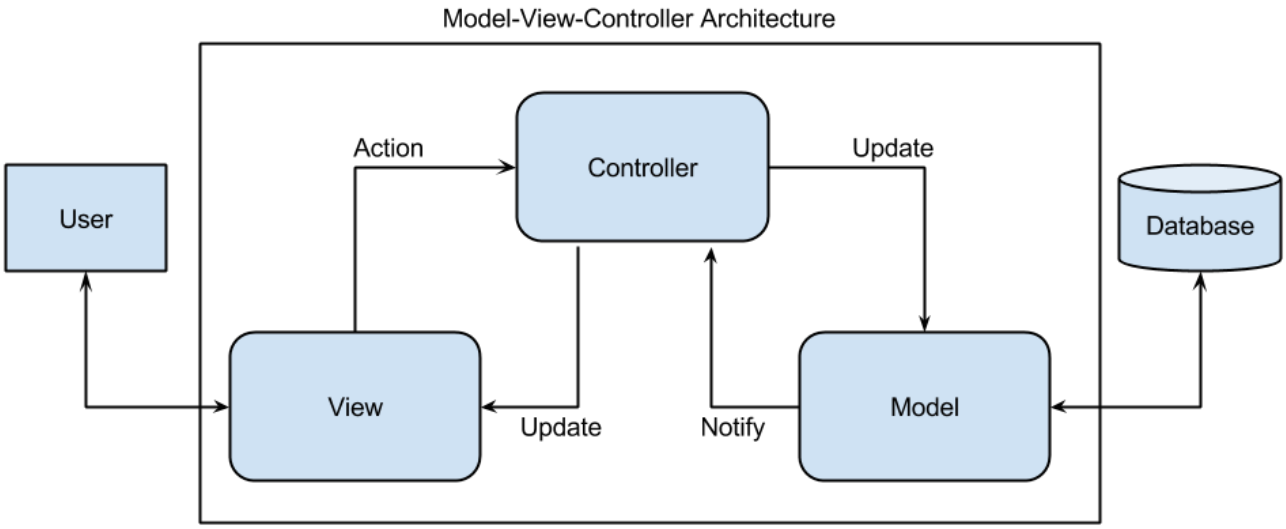
1- **DAO Interface**: Un’ interfaccia che definisce la struttura generale della classe fisica ad essa associata. Essa specifica semplicemente il nome del metodo, il tipo da restituire e i parametri da utilizzare senza specificarne il corpo.
2- **DAO Classe Concreta:** Si intende una classe che implementa l’interfaccia ad essa associata, ciò vuol dire che dovrà obbligatoriamente contenere i metodi specificati nella sua interfaccia importata specificandone però il corpo del metodo. Tale classe ha il compito di estrapolare dati dal database per poi passarli al **Controller**.



Model-View-Controller:

Questo pattern è basato sulla separazione di 3 componenti del pattern:

1-Model
2-View
3-Controller



Model: Il Model ci permette di identificare le funzionalità della piattaforma, in specifico attraverso i metodi che sono racchiusi in ogni classe Model andiamo a interagire con i vari soggetti della piattaforma, andandone a manipolare i dati. Il Model è una componente molto importante, in quanto interagisce direttamente con il Database e viceversa. Tale relazione permette la permanenza dei dati, grazie soprattutto al Database. Ogni volta che si deve manipolare un soggetto dal Database si estrapolano informazioni, e con tali informazioni si va a creare un’entità Model. Viceversa quando si modifica un’entità Model per garantire persistenza dati si aggiorna il Database.

View: La View si occupa di gestire ogni evento all’interno della piattaforma. E’ il package più denso di funzionalità in quanto va a gestire tutte gli eventi della GUI, in particolar modo, ogni volta che l’utente interagisce con la piattaforma, la View ha il compito di mostrare i dati sotto forma di immagini e fa una elaborazione iniziale dei dati che l’utente immette. In un certo senso si può dire che la View fa da tramite tra l’utente e la piattaforma stessa.

Controller: Il controller è un “ponte” che collega la View al Model, per poi finire nel DAO, il quale si occuperà dell’interazione con il Database per la gestione delle varie richieste.
Si occupa anche di controllare che tutto ciò che l’utente immetta sia corretto, in modo tale da garantire una corretta interazione con il Database.

Utilizzando il pattern **MVC**, mantenendo la separazione in 3 stati, è possibile fare una distinzione tra i vari concetti.

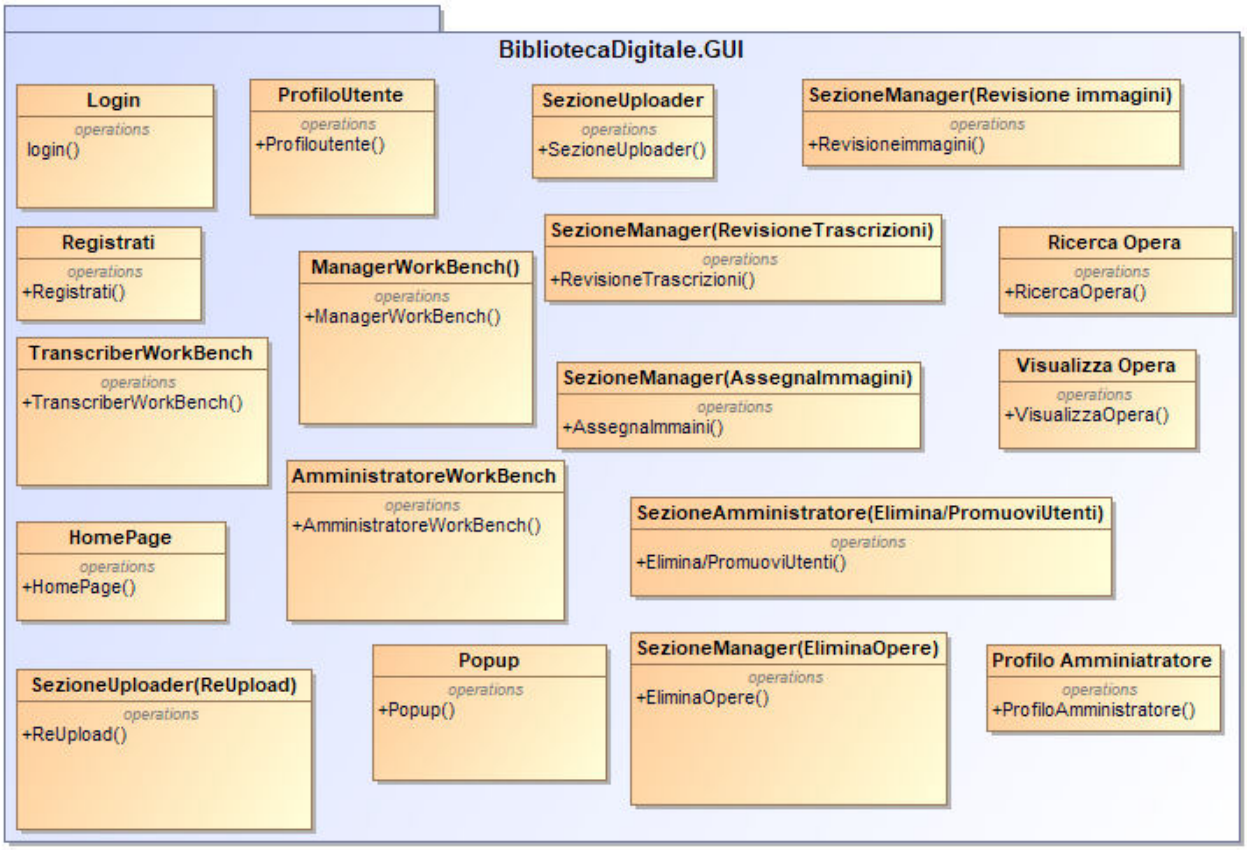
3.0) Software Design:

La piattaforma realizzata è stata separate in diversi package, con lo scopo di mantenere separati i dati presenti nelle varie classi, garantendone una separazione logica. I package individuati sono i seguenti:

- View (contenente anche il package **GUI**);
- Controller;
- Model;
- DAO (contenete anche il package **Database**);

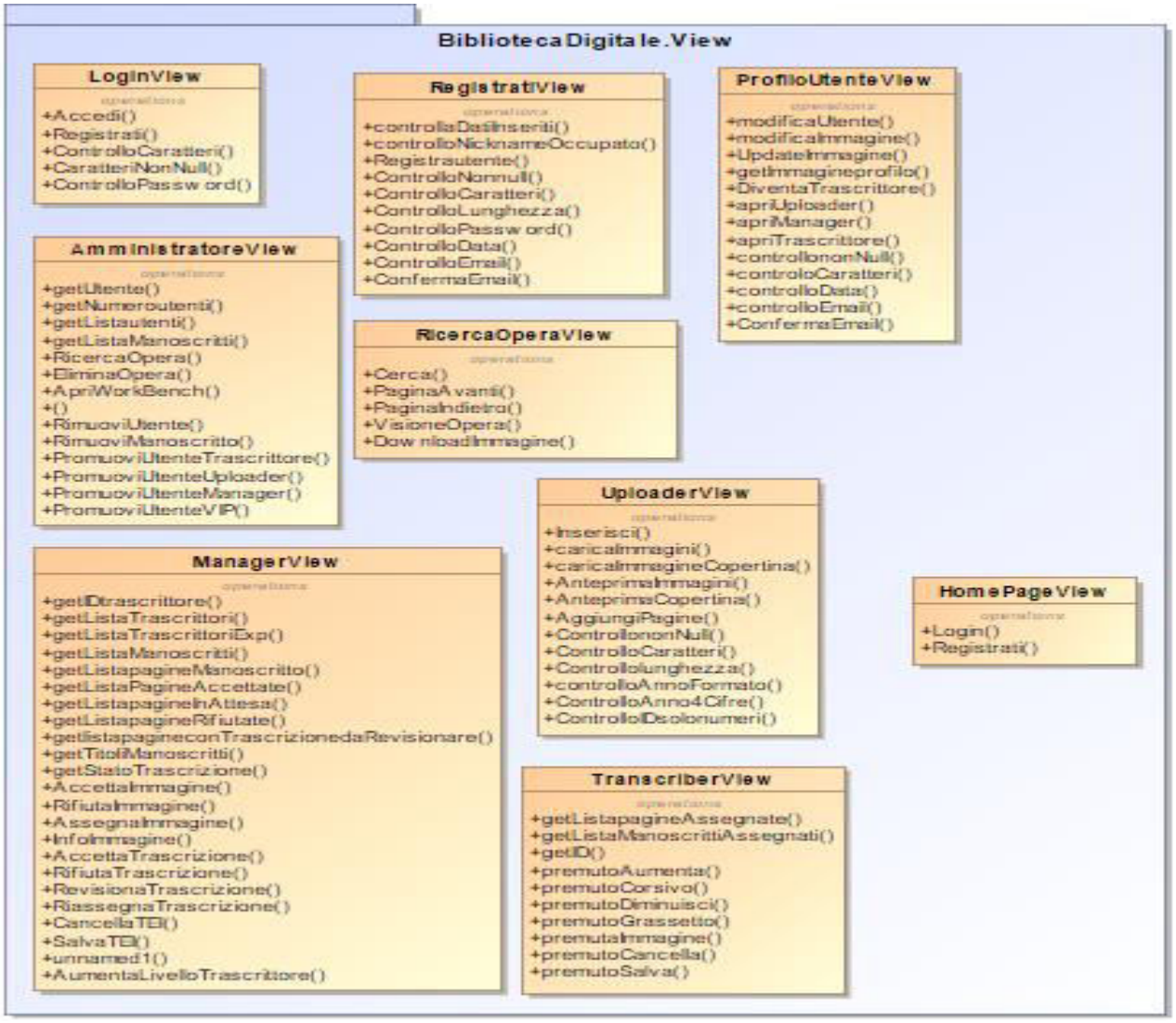
Di seguito sono stati riportati i **Class Diagram** di ciascun package, in cui ne verranno esplicitate le caratteristiche e le varie funzionalità.

Class Diagram – GUI:



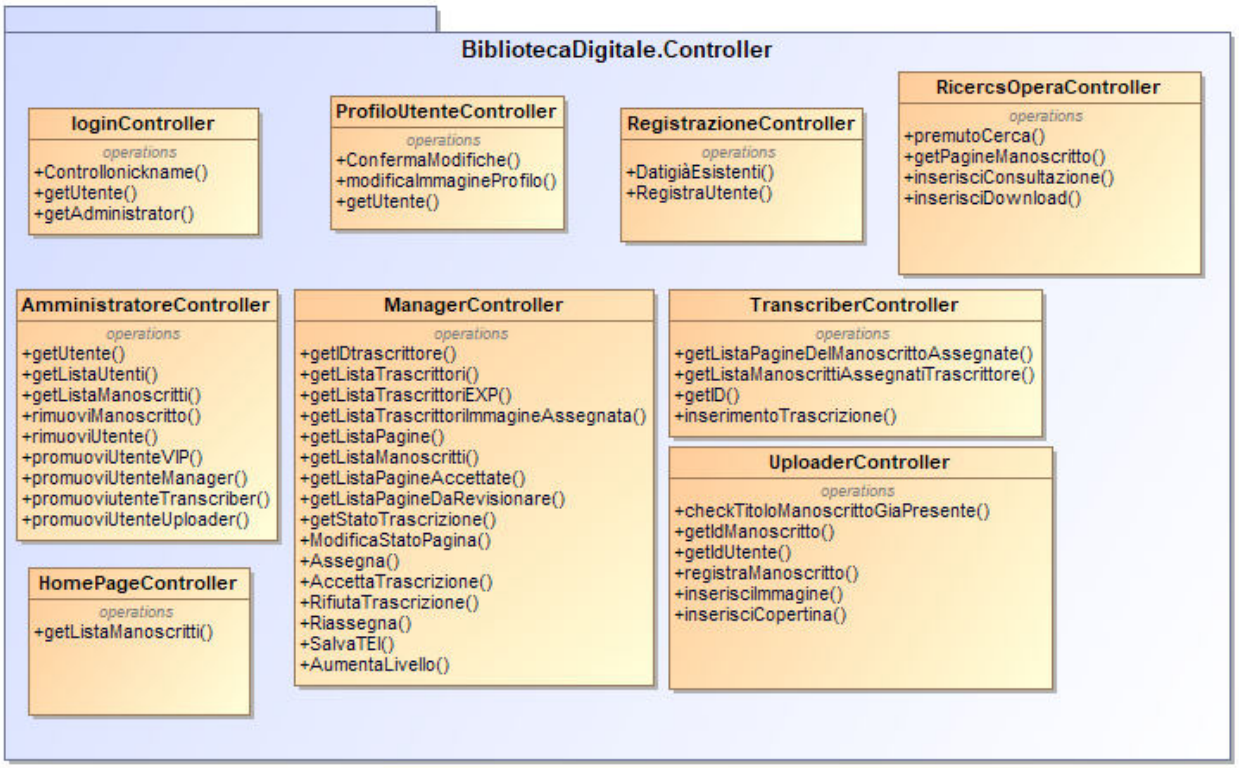
Questo package contiene tutte le GUI (Graphical User Interface) presenti all’interno della piattaforma. Le varie interfacce sono state costruite per mezzo del framework **Swing/AWT** di Java, garantendo un’interfaccia User-Friendly. Tra le GUI sono presenti anche delle finestre di “appoggio” che servono per rendere più facile il lavoro dei vari sottosistemi.

Class Diagram – View:



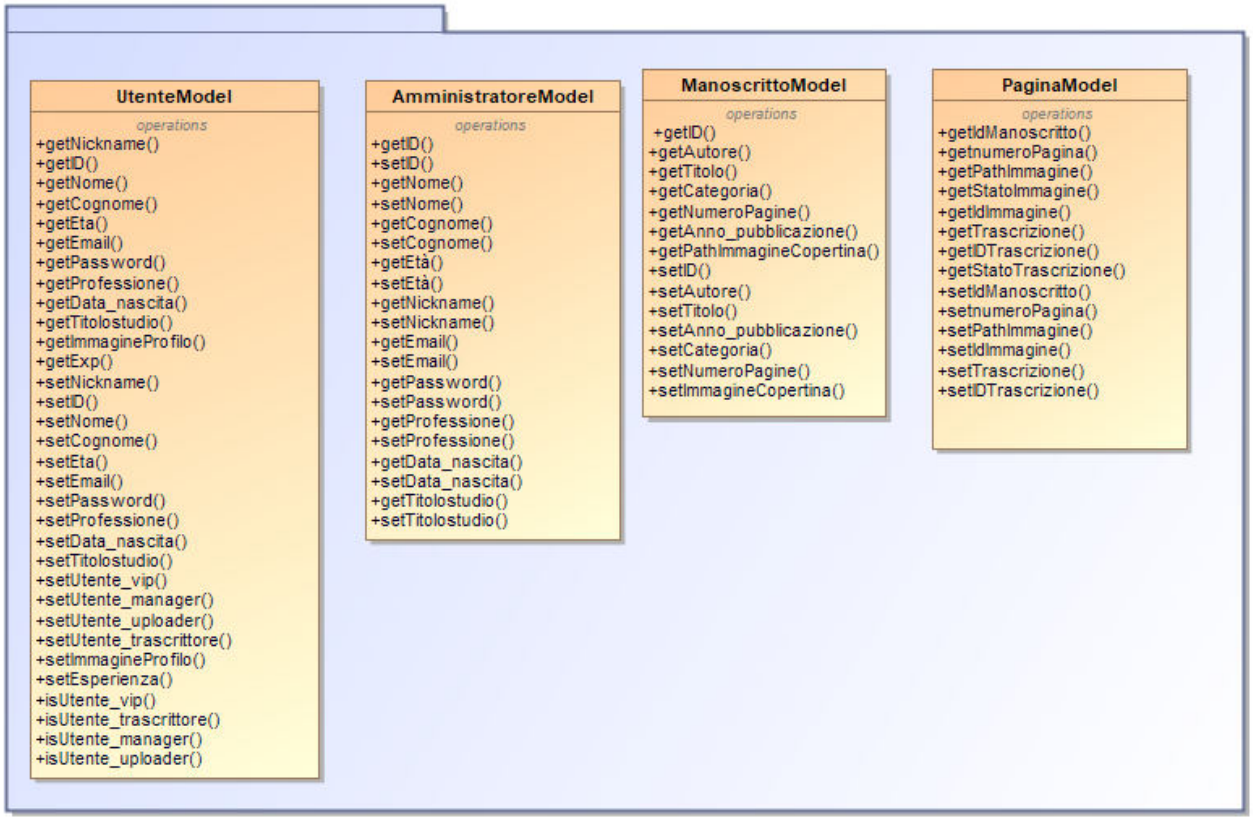
Il package GUI, creato al passo precedente, si avvarrà di metodi presenti nel package View. Una volta fatta questa operazione, la View avrà poi il compito di notificare l’avvenuta ricezione dell’input emesso dall’utente al package Controller, descritto in seguito.

Class Diagram – Controller:



Per ogni classe presente nel package View, avremo una classe corrispondente nel package Controller.

Class Diagram – Model:



Il package Model rappresenta le principali entità presenti nel sistema. Tali classi vengono rappresentate attraverso le principali caratteristiche e funzionalità che le contraddistinguono. All’interno di ogni Classe Model, inoltre, sono presenti costruttori, ed i rispettivi metodi getter/setter.

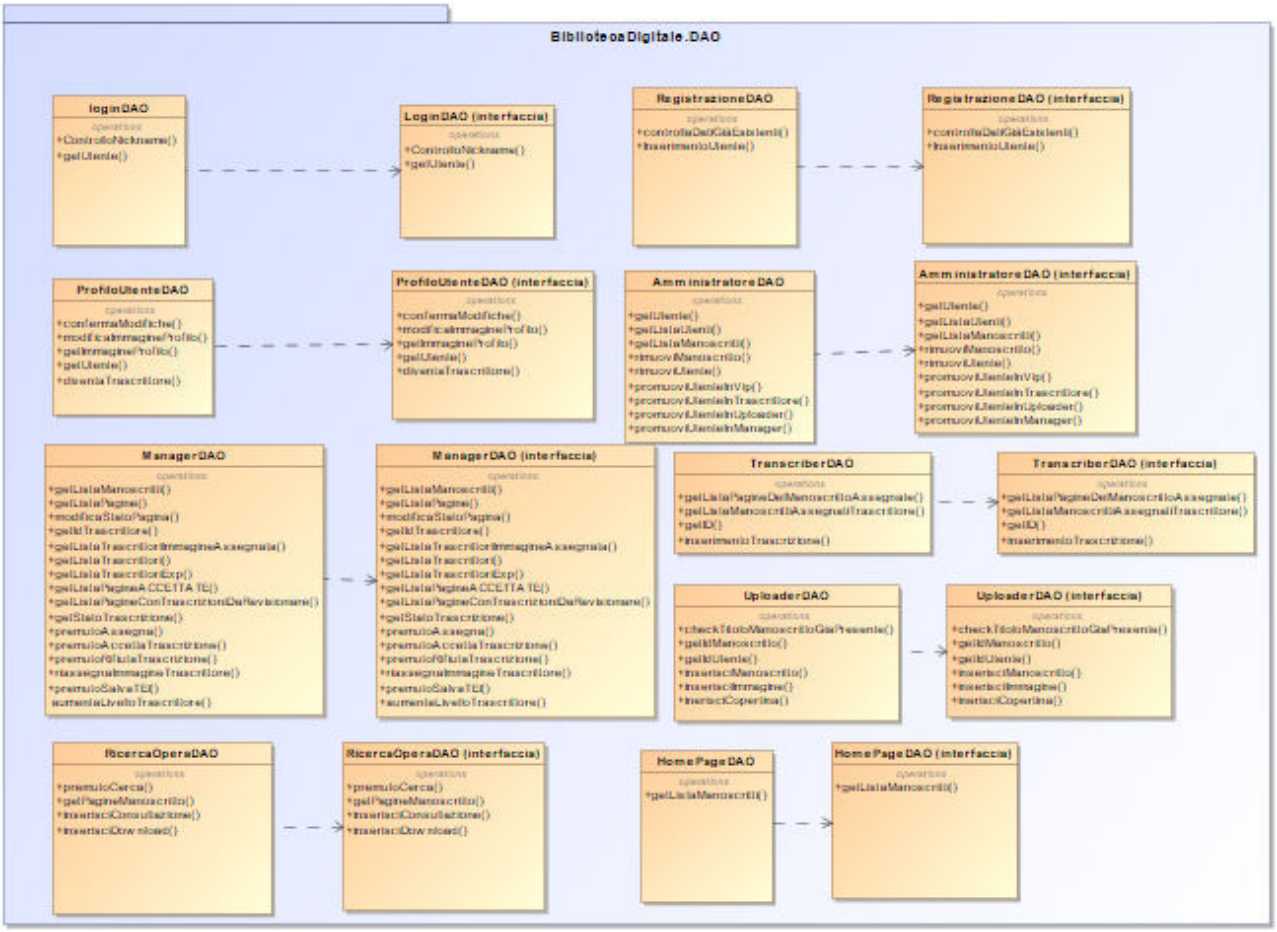
1-**Utente**: L’utente rappresenta la principale entità del sistema, in cui vengono descritte le principali funzionalità e caratteristiche che lo contraddistinguono.

2-**Manoscritto**: Il manoscritto rappresenta principalmente un testo di carattere divulgativo presente all’interno della piattaforma, rappresentato anche esso, cosi come per l’utente, attraverso una serie di informazioni e caratteristiche.

3-**Pagina**: Per pagina si intende, fondamentalmente, il corpo del Manoscritto, in cui figurano un’immagine di una pagina del Manoscritto, con la corrispondente trascrizione affianco.

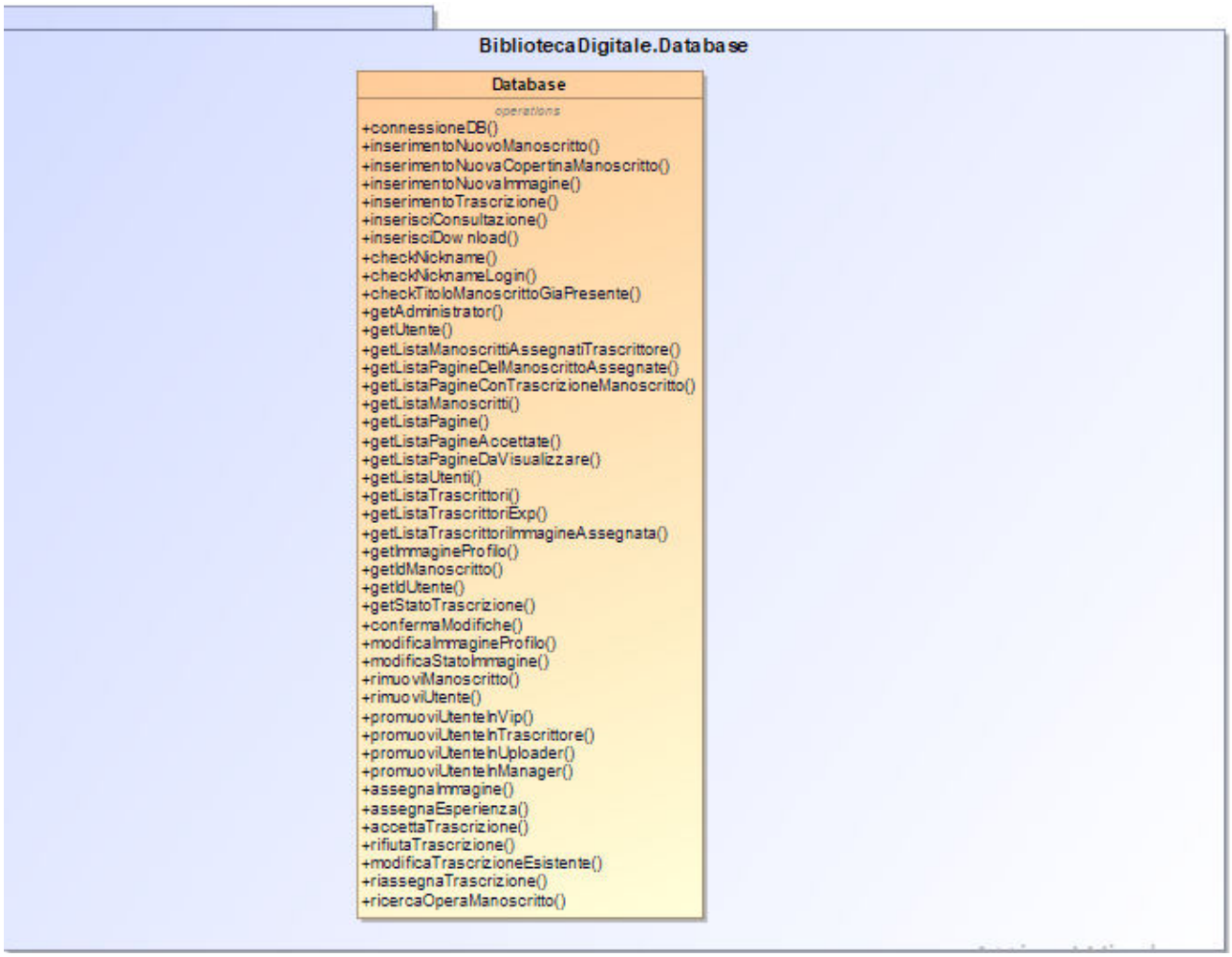
4-**Amministratore**: L’amministratore rappresenta un tipo di Utente, che tuttavia, svolge una sorta di ruolo di supervisore nei confronti dell’intera Piattaforma. Si ha un solo Amministratore per l’intera piattaforma e, come già accennato, egli ha il compito di gestire interamente gli utenti e i manoscritti.

Class Diagram – DAO:

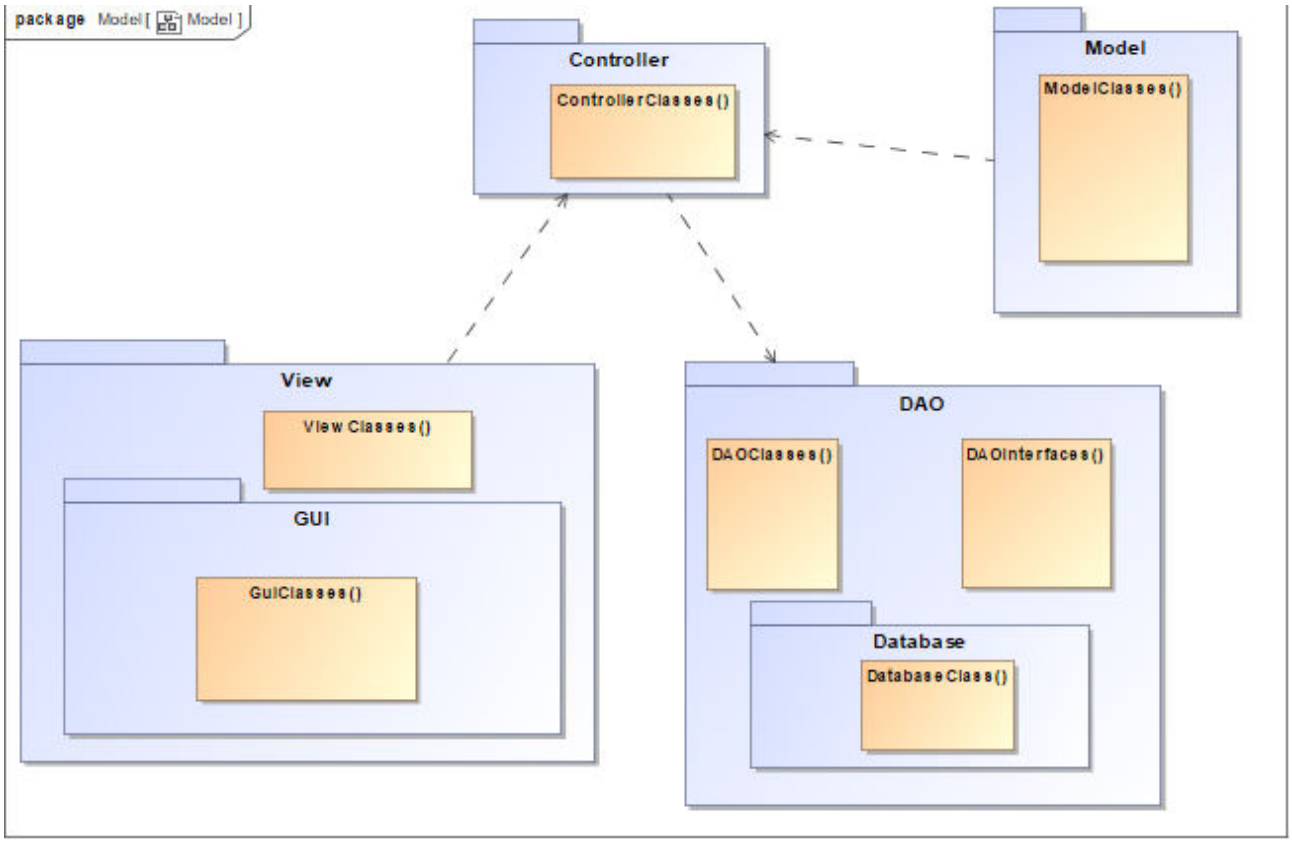


Il DAO rappresenta il principale package di comunicazione con il Database, in cui figurano, oltre alle classi, le rispettive interfacce. Un’interfaccia contiene i metodi (senza corpo) implementati dalle rispettive classi.

Class Diagram – Database:



Dopo una revisione del progetto, e sotto consiglio della Professoressa, si è deciso di modificare l’assettamento dei vari package. Si ottiene così il seguente Class Diagram completo:

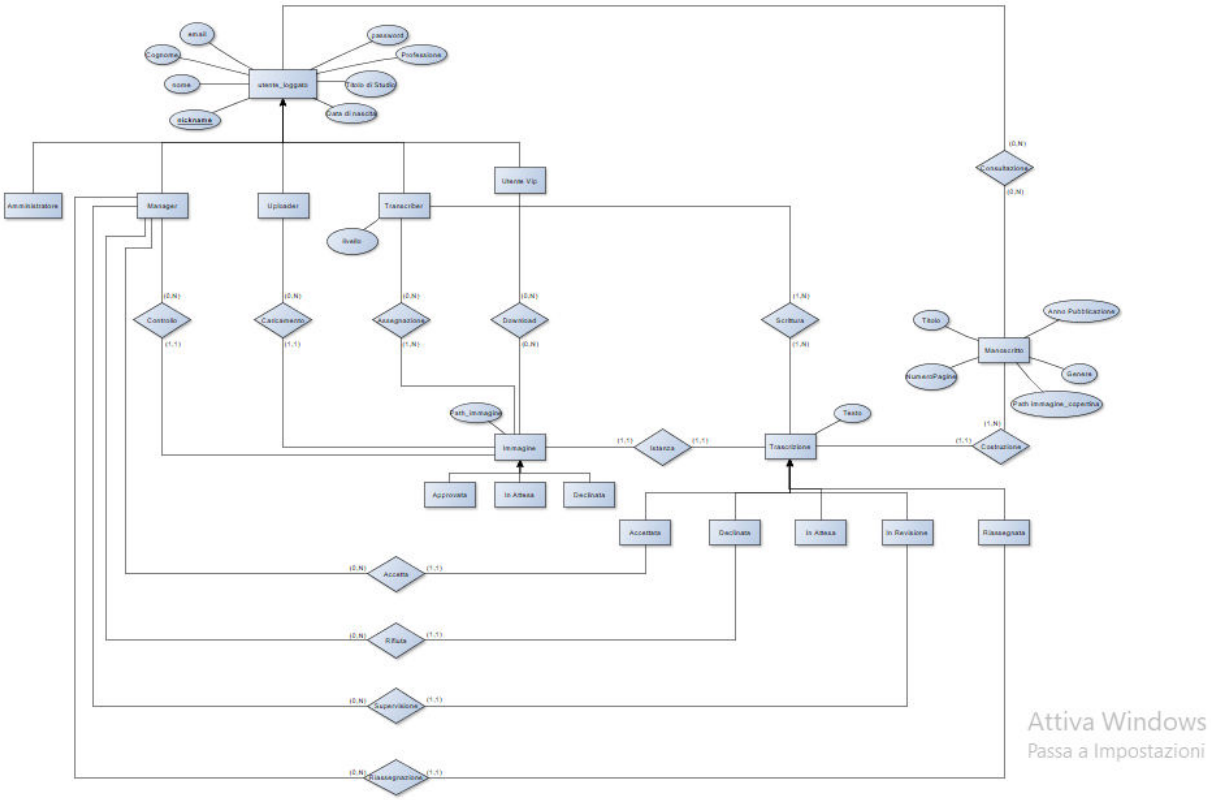


Database:

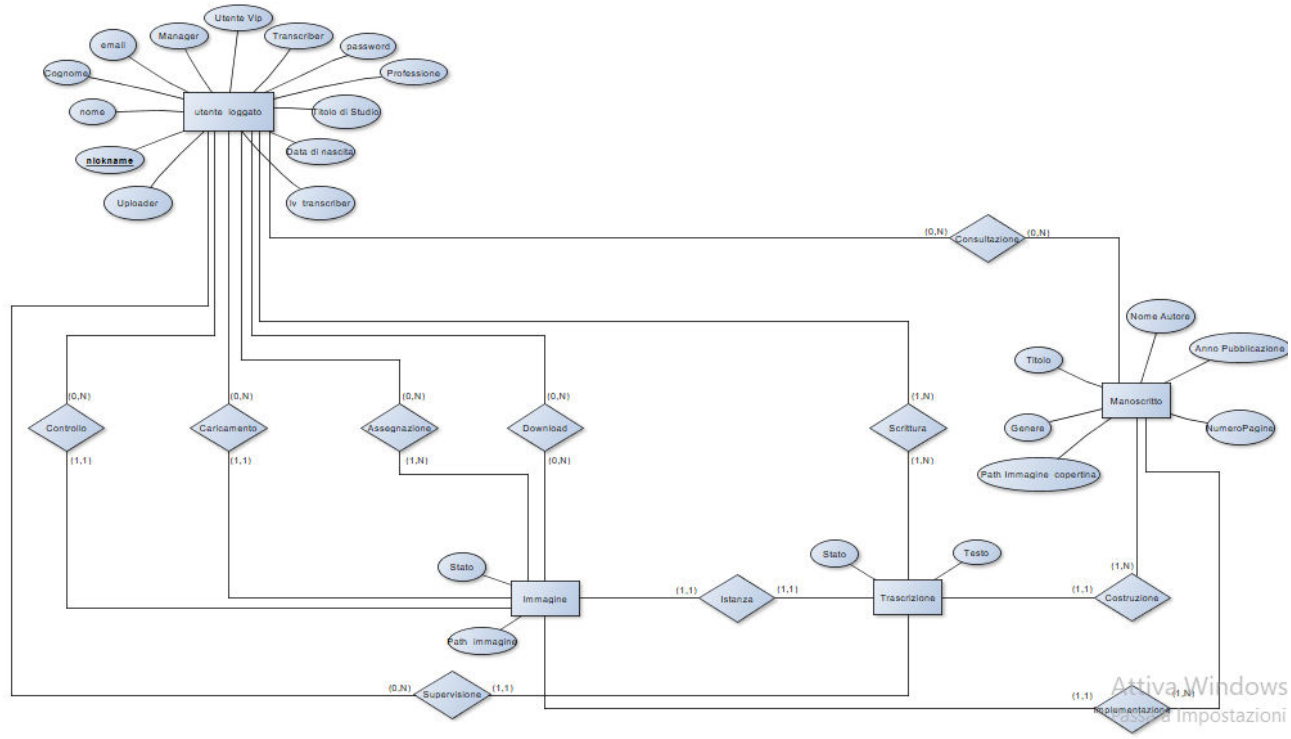
Il database risulta essere una delle componenti fondamentali della piattaforma. Al suo interno è possibile trovare tutte le varie tabelle utili a garantire persistenza dati, riguardanti soprattutto le entità più utilizzate e prese in considerazione come l’utente oppure il Manoscritto. E’ presente anche una tabella contenente tutti i Manoscritti opportunatamente inseriti, con le varie immagini e trascrizioni ad essi associati. La classe Database risulta anche essere la classe più utilizzata. Infatti, considerata la complessità del Database e delle operazioni svolte dai vari sottosistemi della piattaforma, il Database risulta essere aggiornato e interrogato numerose volte, per estrapolare informazioni e per modificare le stesse.

Modello Concettuale:

Seguendo una strategia di tipo “Top-Down” in cui si raggruppano i vari concetti partendo da uno schema “scheletro”, per poi farli “esplodere” in concetti più semplici, si è ottenuto il seguente modello concettuale:



Modello Logico:
Dopo le opportune modifiche ed eliminazioni di concetti presenti nel modello concettuale, come le generalizzazioni di entità, si ottiene alla fine il Modello Logico dei dati. Tale operazione si è resa necessaria in quanto, dal modello concettuale risultante al passo precedente, sono presenti concetti che non sono esprimibili nel Modello Concettuale che si vuole ottenere. Il modello logico risultante è il seguente:



Traduzione verso il Modello Relazionale:

Nel seguente Modello Relazionale sono riportate sia le tabelle sia le relazioni presenti all’interno del modello logico, le quali verranno rappresentate nel Database come normali tabelle.

Utente_loggato (ID, nome, cognome, data_nascita, email, nickname, password, utente_manager, utente_uploader, utente_transcriber,lv_transcriber, utente_vip, titolo_studio, Professione);

Amministratore (ID, nickname, nome, cognome, data_nascita, email, pass_word, professione);

Consultazione (ID_utente, ID_manoscritto);

Manoscritto (ID, titolo, numero_pagine, genere, nome_autore, anno_pubblicazione, path_immaginecopertina);

Immagine (ID, stato, path_immagine, **ID_manager**, **ID_uploader**, **ID_manoscritto**);

Download (ID, **ID_utente**, **ID_immagine**);

Trascrizione (ID, testo, stato, **ID_manager, ID_manoscritto, ID_immagine**);

Assegnazione (ID, **IDimmagine**, **IDtrascrittore**);

Scrittura (ID_trascrittore, ID_trascrizione);

Nello specifico:

-Gli attributi rappresentati sottolineati rappresentano le chiavi primarie delle varie tabelle;
-Gli attributi rappresentati in grassetto rappresentano le **chiavi esterne** delle varie tabelle;

Vincoli di integrità referenziale:

1-Dall’attributo ‘ID’ della tabella **Manoscritto** all’attributo ‘ID_manoscritto’ della tabella **Immagine**;

2-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_uploader’ della tabella **Immagine**;

3-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_manager’ della tabella **Immagine**;

3-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_utente’ della tabella **Download**;

4-Dall’attributo ‘ID’ della tabella **Immagine** all’attributo ‘ID_immagine’ della tabella **Download**;

5-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_manager’ della tabella **Trascrizione**;

6-Dall’attributo ‘ID’ della tabella **Manoscritto** all’attributo ‘ID_manoscritto’ della tabella **Trascrizione**;

7-Dall’attributo ‘ID’ della tabella **Immagine** all’attributo ‘IDimmagine’ della tabella **Trascrizione**;

8-Dall’attributo ‘ID’ della tabella **Immagine** all’attributo ‘IDimmagine’ della tabella **Assegnazione**;

9-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘IDtrascrittore’ della tabella **Assegnazione**;

10-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_utente’ della tabella **Consultazione**;

11-Dall’attributo ‘ID’ della tabella **Manoscritto** all’attributo ‘ID_manoscritto’ della tabella **Consultazione**;

12-Dall’attributo ‘ID’ della tabella **utente_loggato** all’attributo ‘ID_trascrittore’ della tabella **Scrittura**;

13-Dall’attributo ‘ID’ della tabella **Trascrizione** all’attributo ‘ID_trascrizione’ della tabella **Scrittura**;

Creazione del Database e delle tabelle:

Software utilizzato: MySql WorkBench.

1-Creazione del Database:

```
/*Creazione Ddel Database relativo alla Biblioteca Digitale*/
drop database BibliotecaDigitaleOOSD;
create database BibliotecaDigitaleOOSD;
use BibliotecaDigitaleOOSD;
```

2-Tabella utente:

```
create table utente_loggato(
ID integer unsigned not null primary key auto_increment,
nickname varchar(30) not null,
immagine_profilo varchar(300) default null,
nome varchar(200) not null,
cognome varchar(200) not null,
data_nascita date,
email varchar(200) not null,
pass_word varchar(18) not null,
titolo_studio varchar(30) not null,
professione varchar(30) not null,
utente_vip boolean default false,
utente_manager boolean default false,
utente_transcriber boolean default false,
lv_transcriber integer unsigned default null,
utente_uploader boolean default false,
constraint email_unica unique (email),
constraint nickname_unico unique (nickname)
);
```

3-Tabella Amministratore:

```
create table Amministratore(
ID integer unsigned not null primary key auto_increment,
nicknameAmministratore varchar(30) not null default 'Administrator',
nome varchar(200) not null,
cognome varchar(200) not null ,
data_nascita date not null ,
email varchar(200) not null ,
pass_word varchar(18) not null default 'Administrator',
professione varchar(30) not null
);
```

4-Tabella Manoscritto:

```
create table Manoscritto(
ID integer unsigned not null primary key auto_increment,
titolo varchar(200) not null,
immagine_copertina varchar(500) default null,
anno_pubblicazione varchar(4) not null,
numero_pagine integer unsigned not null,
genere varchar(200) not null,
nome_autore varchar(200) not null
);
```

5-Tabella Consultazione:

```
create table consultazione(
ID_utente integer unsigned not null,
ID_manoscritto integer unsigned not null,
primary key (ID_utente,ID_manoscritto),
constraint lettura_utente foreign key (ID_utente) references utente_loggato(ID),
constraint lettura_pagina foreign key (ID_manoscritto) references Manoscritto(ID)
);
```

6-Tabella Immagine:

```
create table immagine(
ID integer unsigned not null primary key auto_increment,
path_immagine varchar(500) not null,
ID_uploader integer unsigned not null,
ID_manoscritto integer unsigned not null,
stato enum('accettata','declinata','In Attesa') not null default 'In Attesa',
ID_manager integer unsigned default null,
constraint manager_immagine foreign key(ID_manager) references utente_loggato(ID),
constraint uploader_immagine foreign key(ID_uploader) references utente_loggato(ID),
constraint manoscritto_immagine foreign key(ID_manoscritto) references manoscritto(ID) on delete cascade on update cascad
);
```

7-Tabella Download:

```
create table download(
ID integer unsigned not null primary key auto_increment,
ID_utente integer unsigned not null,
ID_immagine integer unsigned not null,
constraint download_utente foreign key (ID_utente) references utente_loggato(ID),
constraint download_immagine foreign key (ID_immagine) references immagine(ID)
);
```

8-Tabella Trascrizione:

```
create table trascrizione(
ID integer unsigned not null primary key auto_increment,
testo blob,
stato enum('accettata','declinata','riassegnata','in revisione') not null,
ID_manager integer unsigned default null,
ID_manoscritto integer unsigned not null,
ID_immagine integer unsigned not null,
constraint trascrizione_manager foreign key(ID_manager) references utente_loggato(ID),
constraint trascrizione_manoscritto foreign key(ID_manoscritto) references Manoscritto(ID),
constraint trascrizione_immagine foreign key (ID_immagine) references immagine(ID)
);
```

9-Tabella Assegnazione:

```
create table assegnazione(
ID integer unsigned not null primary key auto_increment,
IDimmagine integer unsigned not null,
IDtrascrittore integer unsigned not null,
constraint assegnazione_immagine foreign key (IDimmagine) references immagine(ID),
constraint assegnazioneimmagine_trascrittore foreign key (IDtrascrittore) references utente_loggato(ID)
);
```

10-Tabella Scrittura:

```
create table scrittura(
ID_trascrittore integer unsigned not null,
ID_trascrizione integer unsigned not null,
primary key (ID_trascrittore, ID_trascrizione),
constraint scrittura_trascrittore foreign key (ID_trascrittore) references utente_loggato(ID),
constraint scrittura_trascrizione foreign key (ID_trascrizione) references Trascrizione(ID)
);
```