

GIANLUCA SCATENA

Matricola 227436

1.0 – Panoramica del sistema

Il progetto prevede lo sviluppo di un software in grado di acquisire una serie di immagini per mezzo di scanner planetari, trascriverne il contenuto e successivamente pubblicarlo. L'idea è quella di un **software utilizzabile dalla propria macchina**, collegato però ad **un database e ad un server esterni**, accessibili da tutti gli utenti registrati.

Ogni immagine, dopo essere stata scannerizzata e acquisita dal sistema, viene prima revisionata (perché potrebbero verificarsi errori di qualsiasi tipo durante il processo di acquisizione) e poi memorizzata. Nella fase di memorizzazione, l'immagine viene categorizzata per uno specifico manoscritto. Infatti, ogni immagine si riferisce esattamente ad una pagina del manoscritto.

Essendo quindi il sistema disponibile a più di un utente, ma potendo decidere allo stesso tempo di caricare una pagina e poi passare ad un'opera completamente differente, è necessario specificare, in fase di caricamento, a quale manoscritto ci si riferisce tra quelli già presenti nel database. Cosa ben più importante riguarda la **consistenza dei dati**, selezionabili e manipolabili soltanto in determinati casi corrispondenti allo stato in cui si trovano.

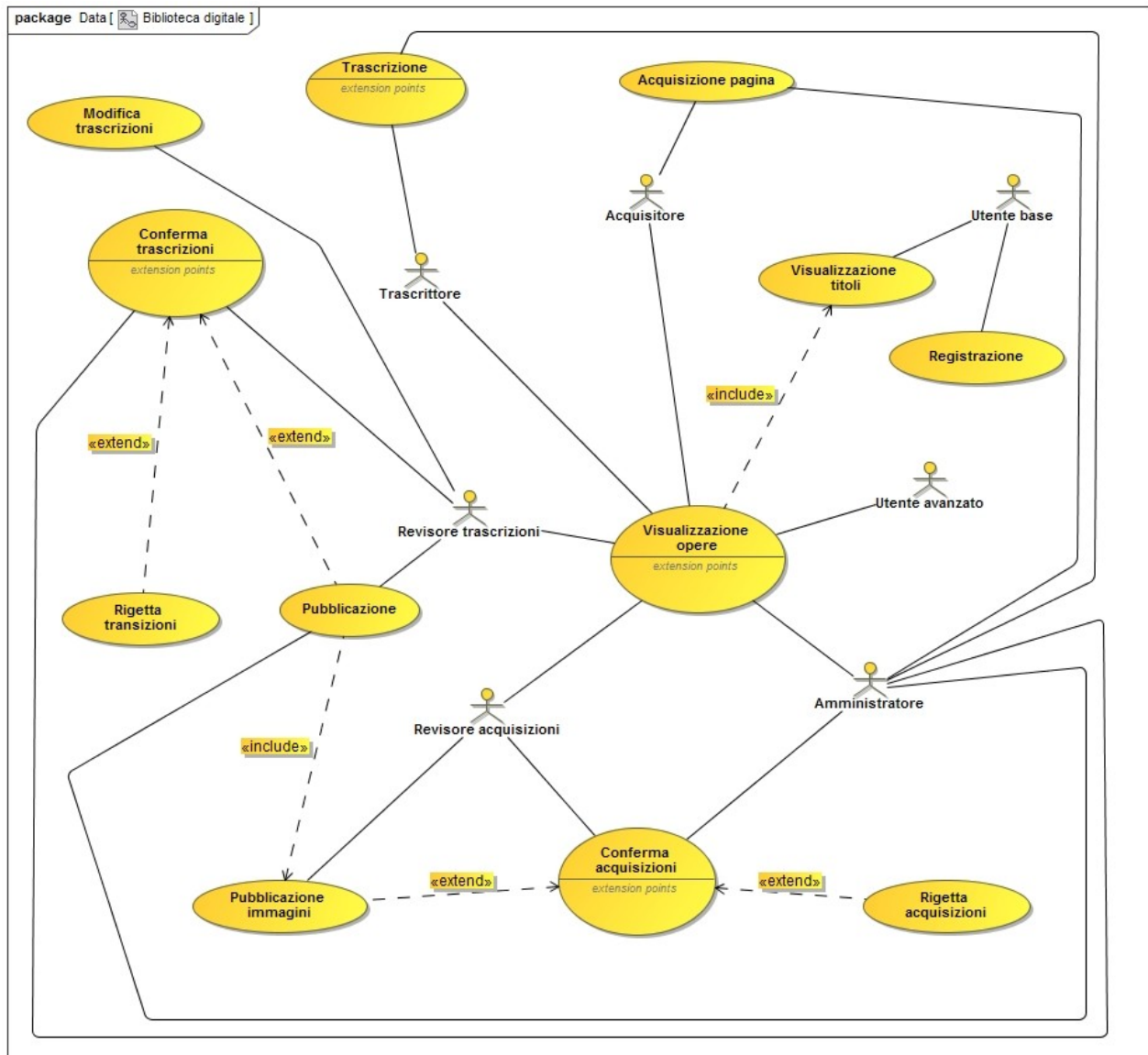
Due sono le tipologie di processo che coinvolgono il sistema:

1) Processo per immagini e trascrizioni: prima della conferma dell'acquisizione dell'immagine, la stessa si trova "*In acquisizione*", passando quindi in "*Acquisito*" dopo la conferma; quando un'immagine viene selezionata per dare inizio alla trascrizione, il dato viene messo "*In trascrizione*", quindi in "*Trascritto*" una volta terminata l'operazione; al contrario, alla fine dell'intero processo, la pagina del manoscritto viene resa pubblica.

2) Processo per immagini: identico al precedente, ad eccezione del fatto che non viene eseguita la trascrizione del contenuto in testo, bensì si esegue direttamente la pubblicazione.

Gli stati suddetti, allo stesso tempo, sono importanti per determinare i ruoli all'interno del sistema e gestire, in particolare, i permessi degli utenti nei confronti dei dati. Ad esempio: un utente non registrato può visualizzare soltanto l'elenco dei titoli, al contrario di un utente avanzato che può visualizzare le opere in dettaglio; in entrambi i casi, tuttavia, ciò che è reso visibile riguarda soltanto i dati pubblicati. Tutto ciò che sta nelle fasi transitorie del processo è manipolabile esclusivamente dagli altri attori del sistema, per ognuno dei quali persistono differenze a livello di permessi.

1.1 - USE CASE DIAGRAM



1.2 – Requisiti in dettaglio

| | |
|----------------------|---|
| Use Case | Acquisizione pagina |
| Attori | Acquisitore, Amministratore |
| Condizioni d'entrata | L'utente ha effettuato l'accesso come Acquisitore |
| Descrizione | Il sistema offre la possibilità di acquisire la pagina di un manoscritto per mezzo di una scannerizzazione, il cui risultato è posto in attesa di revisione dopo la memorizzazione nel database. |
| Codice | AP |
| Scenario | L'utente preme sul bottone relativo all'operazione di acquisizione. Viene effettuata la scannerizzazione per mezzo dell'apposito device e l'immagine viene proposta all'acquisitore come risultato dell'operazione. L'utente decide poi se effettuare nuovamente lo scan oppure se accettare quello già fatto, ponendolo in attesa di revisione. |
| Condizioni di uscita | Una nuova pagina è stata scannerizzata. |

| | |
|----------------------|--|
| Use Case | Conferma acquisizioni |
| Attori | Revisore acquisizioni, Amministratore |
| Condizioni d'entrata | L'utente ha effettuato l'accesso come revisore delle acquisizioni. Devono essere presenti acquisizioni in attesa di revisione all'interno del database. |
| Descrizione | Dopo che l'acquisitore ha effettuato una nuova acquisizione, questa viene messa in attesa di revisione. Il revisore può scegliere in base a quanto proposto dal database, controllare un'immagine e decidere se la scannerizzazione è di qualità accettabile, pubblicandola immediatamente o mettendola a disposizione per una pubblicazione successiva. |
| Codice | CA |
| Scenario | L'utente ha una lista di immagini in attesa di revisione. Ne sceglie una, che successivamente viene visualizzata. Una volta eseguito il controllo, preme sul pulsante di conferma per rendere ufficialmente acquisita la scannerizzazione della pagina. |
| Condizioni di uscita | La scannerizzazione della pagina è acquisita. |
| Esteso da | Pubblicazione immagini, Rigetta acquisizioni |

| | |
|----------------------|--|
| Use Case | Rigetta acquisizioni |
| Attori | Revisore acquisizioni, Amministratore |
| Condizioni d'entrata | L'utente ha effettuato l'accesso come revisore delle acquisizioni. Devono essere presenti acquisizioni in attesa di revisione all'interno del database. |
| Descrizione | Dopo che l'acquisitore ha effettuato una nuova acquisizione, questa viene messa in attesa di revisione. Il revisore può scegliere in base a quanto proposto dal database, controllare un'immagine e decidere se la scannerizzazione è di qualità accettabile. In alternativa, può rifiutare l'acquisizione e richiedere che ne venga effettuata una nuova. |
| Codice | RA |
| Scenario | L'utente ha una lista di immagini in attesa di revisione. Ne sceglie una, che successivamente viene visualizzata. Una volta eseguito il controllo, preme sul bottone per rifiutarla nel caso in cui non soddisfi i criteri di qualità. |
| Condizioni di uscita | Una pagina precedentemente in attesa di revisione viene cancellata dal database e viene notificato all'acquisitore che deve effettuare una nuova acquisizione. |
| Estende | Conferma acquisizioni |

| | |
|----------------------|--|
| Use Case | Conferma trascrizione |
| Attori | Revisore trascrizione, Amministratore |
| Condizioni d'entrata | L'utente ha effettuato l'accesso come revisore delle trascrizioni. Devono essere presenti trascrizioni in attesa di revisione all'interno del database. |
| Descrizione | Dopo ogni nuova trascrizione eseguita da un trascrittore, questa viene messa in attesa di revisione. Il revisore può scegliere in base a quanto proposto dal database, controllare una trascrizione e confermare se lo scritto coincide all'immagine corrispondente. |
| Codice | CT |
| Scenario | L'utente ha una lista di trascrizioni in attesa di revisione. Ne seleziona una, che successivamente viene visualizzata insieme all'immagine relativa. Una volta eseguito il controllo, preme sul bottone per confermarla, potendo poi scegliere se pubblicarla immediatamente o conservarla e pubblicarla in un secondo momento. |
| Condizioni di uscita | Una trascrizione precedentemente in attesa di revisione viene resa disponibile alla pubblicazione. |
| Esteso da | Pubblicazione, Rigetta trascrizioni |

| | |
|----------------------|--|
| Use Case | Rigetta trascrizioni |
| Attori | Revisore trascrizioni, Amministratore |
| Condizioni d'entrata | L'utente ha effettuato l'accesso come revisore delle trascrizioni. Devono essere presenti trascrizioni in attesa di revisione all'interno del database. |
| Descrizione | Dopo ogni nuova trascrizione eseguita da un trascrittore, questa viene messa in attesa di revisione. Il revisore può scegliere in base a quanto proposto dal database, controllare una trascrizione e decidere se il loro contenuto corrisponde o meno all'immagine. Nel caso peggiore, può rifiutare la trascrizione e richiedere al trascrittore che venga modificata, corredando la stessa con opportune note sui punti che richiedono cambiamenti. |
| Codice | RT |
| Scenario | L'utente ha una lista di trascrizioni in attesa di revisione. Ne sceglie una, che successivamente viene visualizzata insieme all'immagine relativa. Una volta eseguito il controllo, preme sul bottone per rifiutarla nel caso in cui non rispetti quanto presentato dall'immagine acquisita nelle fasi precedenti. Scrive poi una serie di note sulle proposte di modifica. |
| Condizioni di uscita | Una trascrizione precedentemente in revisione, viene inclusa in una lista delle trascrizioni da modificare gestibile da un trascrittore. |
| Estende | Conferma trascrizioni |

| | |
|----------------------|---|
| Use Case | Pubblicazione |
| Attori | Revisore trascrizioni, Amministratore |
| Condizioni d'entrata | Una trascrizione è stata confermata e definita corretta. |
| Descrizione | Al termine della revisione, una trascrizione, qualora sia stata considerata corretta, può essere immediatamente pubblicata. La stessa cosa è fattibile in un secondo momento, quando il revisore consulta una lista di trascrizioni già confermate per poi pubblicarle. |
| Codice | PUBB |
| Scenario | L'utente ha una lista di trascrizioni confermate. Ne sceglie una e la rende pubblica, dunque visibile agli utenti del sistema. Automaticamente viene pubblicata anche l'immagine corrispondente al testo, nel caso in cui non sia stata pubblicata in precedenza. |
| Condizioni di uscita | Una trascrizione precedentemente confermata è stata pubblicata. |
| Include | Pubblicazione immagini |
| Estende | Conferma trascrizioni |

| | |
|----------------------|--|
| Use Case | Pubblicazione immagini |
| Attori | Revisore acquisizioni, Amministratore |
| Condizioni d'entrata | Un'acquisizione è stata confermata. |
| Descrizione | Al termine della revisione, un'acquisizione, qualora sia stata considerata corretta, può essere immediatamente pubblicata. La stessa cosa è fattibile in un secondo momento, quando il revisore consulta una lista di acquisizioni già confermate per poi pubblicarle. |
| Codice | IPUBB |
| Scenario | L'utente ha una lista di acquisizioni confermate. Ne sceglie una e la rende pubblica, dunque visibile agli utenti del sistema. |
| Condizioni di uscita | Un'acquisizione precedentemente confermata è stata pubblicata. |
| Incluso in | Pubblicazione |
| Estende | Conferma acquisizioni |

| | |
|----------------------|--|
| Use Case | Trascrizione |
| Attori | Trascrittore, Amministratore |
| Condizioni d'entrata | Un'acquisizione è stata confermata. |
| Descrizione | Al termine della revisione, un'acquisizione, qualora sia stata considerata corretta, può essere presa da un trascrittore, che di fatto ne inizia la trascrizione in un file TEI. |
| Codice | TRAS |
| Scenario | L'utente ha una lista di acquisizioni confermate. Ne sceglie una e ne inizia la trascrizione per mezzo di un editor di testo. |
| Condizioni di uscita | La trascrizione di un'acquisizione viene salvata sul server e messa a disposizione di un revisore. |
| Esteso da | Modifica trascrizioni |

BIBLIOTECA DIGITALE - PROGRAMMAZIONE A OGGETTI 2015/2016

| | |
|----------------------|--|
| Use Case | Modifica trascrizione |
| Attori | Trascrittore, Amministratore |
| Condizioni d'entrata | Una trascrizione è posta in attesa di modifica. |
| Descrizione | Quando uno degli amministratori del sistema (consultando le opere in maniera indipendente oppure seguendo una delle segnalazioni degli utenti) invia richiesta di modifica per una trascrizione, questa appare nella lista delle trascrizioni da modificare, dalla quale il trascrittore può attingere ed eseguire le modifiche. |
| Codice | MODTRAS |
| Scenario | Il trascrittore apre la lista delle trascrizioni in attesa di modifica. Ne seleziona una ed inizia la “nuova” trascrizione per mezzo dello stesso editor di testo. La vecchia trascrizione viene poi sovrascritta. |
| Condizioni di uscita | Una trascrizione è stata modificata e viene posta nuovamente in revisione. |
| Estende | Trascrizione |

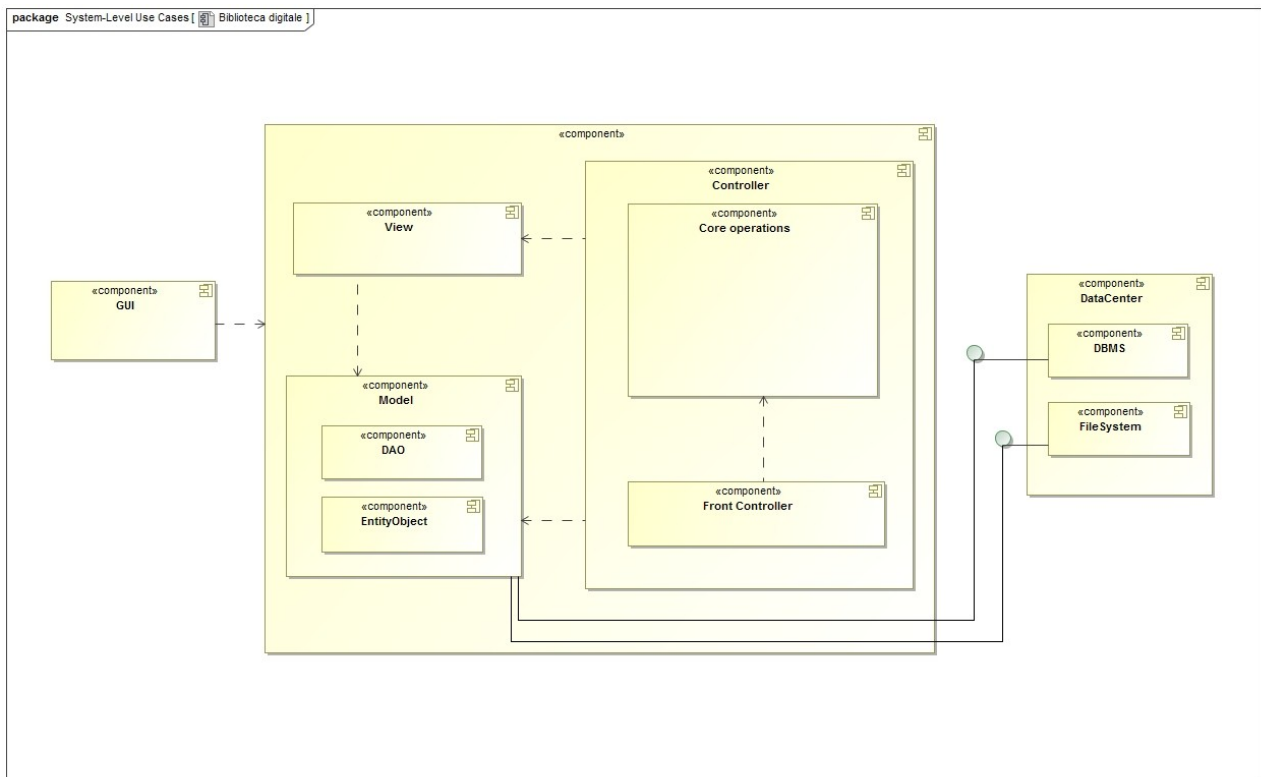
| | |
|----------------------|--|
| Use Case | Visualizzazione titoli |
| Attori | Utente base |
| Condizioni d'entrata | L'utente loggato non ha privilegi particolari. |
| Descrizione | Viene visualizzata la lista dei titoli di tutte le opere caricate nel sistema. |
| Codice | VT |
| Scenario | L'utente effettua il login ed è limitato alla visualizzazione di una lista dei titoli di tutte le opere. |

| | |
|----------------------|---|
| Use Case | Visualizzazione opere |
| Attori | Tutti tranne utente base |
| Condizioni d'entrata | L'utente loggato non è un utente base. |
| Descrizione | L'utente non è più limitato alla semplice visualizzazione di una lista di titoli, ma cliccandovi sopra ottiene l'opera in dettaglio. |
| Codice | VO |
| Scenario | L'utente effettua il login ed ha una lista dei titoli di opere. Clicca su uno di questi e ne ottiene i dettagli (lista di pagine). |

BIBLIOTECA DIGITALE - PROGRAMMAZIONE A OGGETTI 2015/2016

| | |
|----------------------|---|
| Use Case | Registrazione |
| Attori | Utente base, Amministratore |
| Condizioni d'entrata | L'utente non è iscritto al sistema. |
| Descrizione | L'utente base ha un raggio d'azione limitato, per cui gli viene data la possibilità di registrarsi, così da avere privilegi ulteriori e visualizzare le opere in dettaglio. |
| Codice | REG |
| Scenario | L'utente base scorre la lista dei titoli delle opere, ma non può visualizzarle nel dettaglio. Decide quindi di registrarsi, compilando i campi del form. |
| Condizioni di uscita | Un nuovo utente avanzato è stato creato nel database. |

2.0 – COMPONENT DIAGRAM



2.1 – Descrizione dell'architettura del software

Il sistema basa il suo lavoro su tre componenti distaccate. La prima coincide con l'interfaccia grafica utilizzata dall'utente, *in collaborazione* con la parte di controllo (e non solo) in cui vengono gestite tutte le operazioni disponibili. L'altra parte riguarda invece i dati manipolati durante l'utilizzo della piattaforma (lato server).

Nello specifico, l'interfaccia grafica fa uso di un visualizzatore dettagliato delle opere, o più semplicemente di una lista delle opere memorizzate. Questa parte fa uso di una componente **DAO (Data Access Object)** che gestisce tutte le richieste al database o al filesystem del server.

La **GUI**, dunque, presenta una disposizione grafica nella quale risaltano le operazioni eseguibili in relazione ai privilegi dell'utente e alla specifica visualizzazione.

Tracciabilità con requisiti: [VT, VO, NOT, GS]

Queste operazioni ricorrono ad un'unica componente, **Front controller**, che fa da supervisore di tutte le richieste “volontarie” dell'utilizzatore. È di fatto un collegamento tra la GUI e le principali funzionalità del sistema.

Dovendo sostanzialmente gestire i dati, le singole componenti che implementano le funzioni sono raggruppate in un unico sottosistema che si appella alla suddetta interfaccia verso il server esterno, il cui ruolo è principale riferimento per tutti i dati richiesti, ma più in generale rispecchia il fulcro dell'intera piattaforma.

In questo contesto, un ruolo fondamentale è svolto in aggiunta dal membro **EntityObject**, contenente tutti gli oggetti modellati attorno ai dati scambiati.

In dettaglio, il controller presenta un sottocomponente che fornisce le funzionalità cardine dell'intero sistema:

- **Core operations:** gestisce le operazioni relative al salvataggio di una trascrizione, che viene convertita in un testo XML TEI, e all'acquisizione di nuove immagini, avvalendosi della semplice operazione di upload del file.

Tracciabilità con requisiti: [AP, TRAS, MODTRAS]

Si occupa inoltre di tutte le operazioni rivolte al cambiamento di stato a cui la pagina di un'opera è sottoposta nel processo di pubblicazione e controlla eventuali nuove registrazioni.

Tracciabilità con requisiti: [CA, CT, RT, RA, REG, GU, PUBB, IPUBB]

2.2 – Descrizione delle scelte e delle strategie

- Perché MVC?

Il design pattern Model-View-Controller è la massima espressione della *Separation of Concerns*. Permette infatti di suddividere la presentazione dalla logica del sistema, sposando a tutti gli effetti l'idea dell'architettura precedentemente sviluppata.

La struttura è stata pensata per dividere la view, utilizzata come semplice raccolta delle funzionalità messe a disposizione degli utenti, dal resto del software, dato da una parte di controllo sulle azioni richieste dall'utente e da altre due componenti in cui i dati vengono manipolati.

Il vantaggio di questa scelta sta in un buon livello di manutenibilità, dovuto al fatto che ogni sottosistema lavora in modo indipendente, prendendo dagli altri soltanto il risultato delle funzioni senza tener conto di come queste vengano implementate. Un effetto che permette quindi di lavorare soltanto su piccole parti del sistema qualora ci sia il bisogno di effettuare modifiche. Da qui, è facile spiegare perché entrano in gioco le componenti aggiuntive relative a DAO, Front Controller ed EntityObject.

- Perché Front Controller, DAO e EntityObject?

Tramite l'interfaccia grafica, sono mostrate all'utente le opere caricate nel sistema, ma al contempo tutte le operazioni che possono essere effettuate su di esse o in generale grazie al software.

Le operazioni fanno riferimento ai componenti implementati nel sottosistema di controllo. Tuttavia, collegando ogni operazione singolarmente ad ognuno di questi componenti, si otterrebbe un effetto indesiderato dovuto ad un elevato grado di coupling. Da ciò, una delle conseguenze principali sarebbe stata l'elevata dipendenza della GUI nei confronti dell'implementazione delle funzionalità.

Con l'introduzione di un Front Controller, questo effetto viene evitato, dal momento che si raccolgono in un singolo componente tutte le richieste provenienti dall'interfaccia grafica, successivamente elaborate e “reindirizzate”.

Alla luce dell'effettivo ruolo del sistema, incentrato sull'incessante connessione con il mondo dei dati, stesso ragionamento è stato intrapreso per introdurre le classi DAO ed EntityObject, atte a facilitare lo scambio di dati con il server esterno. Tali classi permettono infatti di modellare oggetti sui dati richiesti o inviati, rendendo la gestione di questi indipendente dal lavoro svolto in fase di esecuzione delle query. In base alle richieste pervenute, la classe DAO effettua le query al database o preleva files consultando il filesystem del server esterno in maniera del tutto oscura al resto del sistema, rispettando quindi i criteri di information hiding e fornendo di conseguenza nuovi oggetti ispirati e plasmati sui dati ricavati.

- Perché i dati sono su un server esterno?

L'architettura progettata evidenzia la reale natura del software, il quale si suddivide ideologicamente in due parti: una dedicata all'interfaccia grafica e al controllo delle funzioni, l'altra all'immagazzinamento dei dati. La particolarità del software sta nel fatto che può essere visto come un'applicazione web, sviluppata però come applicazione desktop.

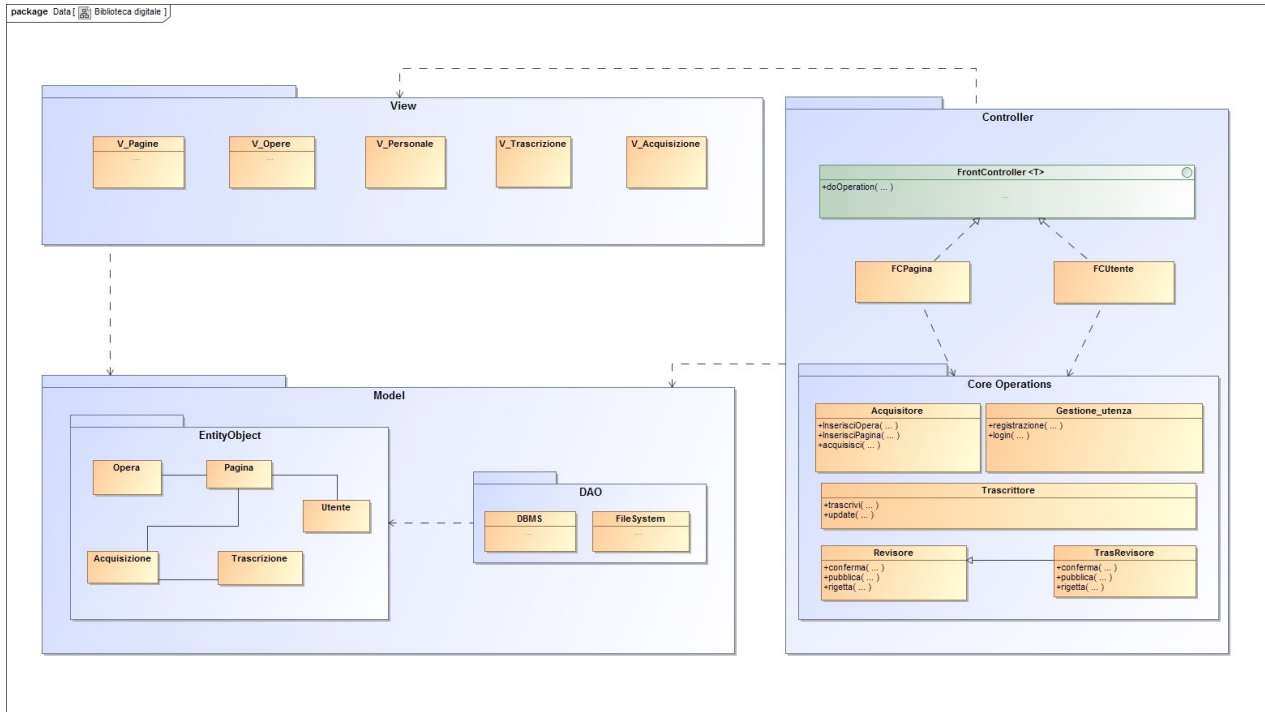
Le due parti che compongono la struttura si collocano in luoghi del tutto diversi geograficamente: GUI e controllo vengono gestiti in locale, mentre i dati sono immagazzinati in un server esterno, accessibile da più utenti contemporaneamente. Si deduce infatti che sulle opere possano lavorare più utenti tra loro distanti, riportando poi le “modifiche” ed il lavoro svolto direttamente sul server, in modo che sia visibile a tutti coloro che utilizzano il software nella propria macchina.

In questo senso, inoltre, sono state prese decisioni importanti per mantenere la consistenza dei dati, modificabili soltanto da una persona per volta.

Con un occhio di riguardo alla consistenza, si è pensato di specificare lo stato di un dato all'interno del database, in base al quale è possibile effettuare determinate operazioni per mezzo di controlli preventivi (se ad esempio una pagina già acquisita fosse in trascrizione da un utente, nessun altro potrebbe prenderla per effettuare una nuova trascrizione, ma dovrebbe aspettare la fine di quella attuale). Vedendo gli stati in dettaglio:

- *In acquisizione*: la pagina del manoscritto sta per essere scannerizzata da un Acquisitore;
- *In attesa di revisione*: la pagina è stata scannerizzata ed è in attesa che un revisore la selezioni;
- *In revisione acquisizione*: l'immagine è in fase di approvazione da un Revisore acquisizioni;
- *Acquisito*: l'immagine è stata confermata da un Revisore acquisizioni, il quale può in seguito decidere di pubblicarla o meno. In ogni modo, nel caso in cui l'immagine venisse pubblicata, risulterebbe comunque acquisita. Di contro, quando si chiede di tornare ad uno stato precedente, automaticamente viene dismessa la pubblicazione;
- *In trascrizione*: l'immagine è stata presa da un Trascrittore che si occuperà di digitalizzarne il contenuto;
- *In attesa di revisione*: il contenuto è stato digitalizzato e deve essere revisionato;
- *In revisione trascrizione*: al termine della trascrizione, si aspetta la conferma di un Revisore trascrizioni;
- *Trascritto*: la trascrizione è stata confermata da un Revisore trascrizioni, che come il precedente può pubblicare o meno;
- *Pubblicato*: la digitalizzazione della pagina è resa disponibile agli utenti.

3.0 - CLASS DIAGRAM



3.1 – Descrizione dei dettagli di design

Nel class diagram viene rappresentato il sistema indipendentemente da come questo si interfaccia al server esterno. Viene quindi preso in considerazione il macro componente utilizzato in macchina locale (alla luce dell'architettura descritta). Da qui nascono alcune delle scelte di design.

Innanzitutto, le classi che implementano il software sono state organizzate in packages che rispettano la struttura precedentemente mostrata dal Component Diagram. Trattandosi di una semplice collezione di dati da dover mostrare all'utente (in base al pattern MVC), nel package relativo alla **View** troviamo un insieme di classi il cui obiettivo è gestire liste di opere, pagine, oppure singole acquisizioni/trascrizioni, richiedendole direttamente alla classe **DAO** e permettendo allo stesso utente di decidere quali azioni effettuare e su quali entità, per mezzo dell'interfaccia grafica che quindi dipende dalla View.

Analizzando un ipotetico scenario di utilizzo si ricorre a:

- *getOpere()*: viene chiamata inizialmente per richiedere la lista di tutte le opere presenti nel database; queste vengono passate per mezzo di una collezione di oggetti *Opera*, contenenti (ad esempio) soltanto il nome.

- *get (Opera)*: è un metodo polimorfo al quale viene passato un oggetto **Opera** preso dalla lista precedente. In base all'oggetto passato, quindi, la classe DAO esegue una specifica query per richiedere l'opera in dettaglio, corredata da una collezione di pagine acquisite, modificando quindi l'oggetto precedente (per questo void).

Successivamente a questi due metodi appartenenti alla classe DAO, possono essere utilizzati quelli della View per filtrare la lista di pagine/opere (in base anche ai privilegi).

In questo senso, quindi, svolge un ruolo fondamentale il package in cui sono raggruppate tutte le classi che modellano i dati scambiati col database (**EntityObject**).

Le principali operazioni offerte all'utente vengono invece rappresentate da una singola interfaccia (**FrontController**), nella quale è definito un metodo che fa da ponte alle vere e proprie operazioni eseguite a livello di controllo. Questa decisione è stata presa prima di tutto per facilitare la gestione e la manutenibilità delle operazioni nella GUI, laddove sarà possibile giostrarle chiamando sempre lo stesso metodo eseguito su un parametro.

Di fatto, il metodo si aspetta un oggetto **HashMap**, in cui l'operazione da effettuare coincide alla chiave, mentre un array (di argomenti) è il valore associato alla chiave.

Questa tecnica permette di isolare completamente la GUI dal resto del programma,

rispettando uno dei criteri fondamentali della manutenibilità. Se ad esempio si volesse un giorno modificare l'implementazione interna delle operazioni, non ci sarebbe sicuramente bisogno di intervenire su tutte le classi del software.

Come accennato in precedenza, le classi del DAO si appoggiano evidentemente sull'importanza del polimorfismo, definendo sostanzialmente due metodi: *get(...)* e *set(...)*, entrambi definiti in base all'argomento ricevuto. Il primo svolge le query per ottenere ulteriori informazioni dal database, il secondo invece per aggiornarne i records.

Riassumendo, quindi, i due obiettivi che mi hanno portato a sviluppare le scelte di design descritti sono:

- **Coinvolgere il minor numero di dati possibile** nello scambio con il server e allo stesso tempo non rallentare la visualizzazione delle opere gestendo, dove possibile, le opere e le pagine in locale.
- **Accentuare il più possibile la separation of concerns** in modo tale da generare operazioni non del tutto dipendenti dal resto del sistema, ottenendo dunque un buon grado di manutenibilità.

4.0 – DATABASE: SCHEMA E-R

