Descrizione dei dettagli di Design scelti

Legenda:

Classe

Variabile istanza

Affermazioni importanti

1. Model

La classe **Document** include dentro sé tutte le informazioni riguardanti l'opera, le pagine ed i relativi progetti di digitalizzazione e trascrizione.

Nella classe **Document** infatti troviamo un oggetto **metadata** che incapsula le informazioni che caratterizzano l'opera, un oggetto **ScanningWorkProject** che include dentro sé tutti gli oggetti riguardanti il progetto di digitalizzazione dell'opera e una classe **TranscriptionWorkProject** che riassume tutte le informazioni riguardanti il progetto di trascrizione. Essendo l'opera costituita da un numero variabile di pagine, essa include dentro una lista di riferimenti ad oggetti di classe Page. Inizialmente un nuovo Document contiene solamente i metadati.

Per associare delle pagine al **Document** è necessario creare un nuovo progetto di digitalizzazione.

La classe **WorkProject** rappresenta un generico progetto. Essa comprende un attributo *coordinator* che rappresenta il responsabile del progetto ed un attributo *PublishingDate* che rappresenta la data di pubblicazione del progetto. Di default la data è inizializzata a null stando ad indicare che la relativa digitalizzazione o trascrizione non è stata ancora pubblicata, solo quando la data verrà inserita il lavoro sarà visibile al generico utente registrato.

Le classi **ScanningWorkProject** e **TranscriptionWorkProject** sono estensioni della classe **WorkProject** e si differenziano per le loro applicazioni.

Un progetto di digitalizzazione prevede la creazione di uno **ScanningWorkProject** da parte di un coordinatore e la scelta di una lista di digitalizzatori e di revisori.

I digitalizzatori creano nuove **Page** da aggiungere al **Document**. La classe **Page** incapsula al suo interno un oggetto della classe **PageScan** che rappresenta la scansione di quella pagina e una classe **PageTranscription** che rappresenta la trascrizione di quella pagina. Il digitalizzatore oltre alla **Page** crea anche la relativa **PageScan** che contiene l'immagine della scansione.

Le classi **PageScan** e **PageTranscription** sono dotate delle rispettive liste ***PageScanStaff*** *e* ***PageTranscriptionStaff*** che rappresentano il personale che ha lavorato a quella particolare scansione o trascrizione. Inoltre queste due classi possiedono dei flag per segnalare lo stato della revisione e l'esito della revisione. Il digitalizzatore può riordinare le **Page** cambiando il loro numero di pagine (*Page Number*), può anche eliminare e ricaricare scansioni fino a quando tutto il lavoro non è completato.

Quando i digitalizzatori hanno terminato l'intera scansione dell'opera possono aggiornare l'attributo *completed* dell'oggetto **ScanningWorkProject**, solo dopo il caricamento di tutte le immagini i revisori potranno incominciare la revisione delle scansioni, ogni singola **PageScan** possiede un attributo *revised* per

indicare se è stata revisionata ed un attributo *validated* per indicare se è stata validata da un revisore. Quando tutte le scansioni sono state validate allora l'opera potrà essere pubblicata dal coordinatore impostando automaticamente la data di pubblicazione nello **ScanningWorkProject**.

Un progetto di trascrizione prevede la creazione di un **TranscriptionWorkProject** da parte di un coordinatore sulla base di una scansione delle pagine già esistente e la scelta di una lista di trascrittori e di revisori. La classe **PageTranscription** possiede un attributo **TEIText** ed un attributo **PageScanStaff** che rappresenta il personale che ha lavorato alla singola trascrizione. Inoltre sono presenti degli attributi che segnalano lo stato della trascrizione, lo stato di revisione e l'esito della revisione. I trascrittori del progetto accedono in maniera mutualmente esclusiva alle trascrizioni delle pagine, quando una singola trascrizione viene completata viene settato l'attributo *completed* consentendone l'accesso ai revisori.

Quando un revisore termina la revisione cambia il valore dell'attributo *revised* (che indica il completamento della revisione) e quello dell'attributo *validated* (che specifica l'esito della revisione)

La trascrizione dell'intero **Document** è pubblicabile quando tutte le revisioni di ogni trascrizione sono state validate. La pubblicazione avviene impostando la data di pubblicazione.

La classe **DocumentCollection** rappresenta una raccolta di **Document** (opere).

La classe **SessionDataHandler** si occupa di gestire gli oggetti caricati dall'applicazione e di far corrispondere ad essi i corrispondenti **UUID**, gli **UUID** permettono di rappresentare oggetti all'interno di altri oggetti.

Le classi **User** e **UserInformations** rappresentano l'utente e le relative informazioni.

2. Controller

2.1 La classe **SessionDataHandler**[Controller] fa da contenitore per tutti gli oggetti del Model. **SessionDataHandler** incapsula una **HashMap**<**UUID**,**Object**>, ogni accesso ai membri della HashMap avviene attraverso metodi specifici per le varie classi Model (Page, Document , ecc.). **SessionDataHandler** si interfaccia con le classi del DAO per caricare automaticamente in memoria gli oggetti richiesti non presenti nella HashMap e rimuovere quelli non più necessari.

Nelle classi del Model ogni riferimento ad altri oggetti Model è contenuto come **UUID**, fornendo però dei metodi per restituire i riferimenti diretti in memoria (che vengono forniti attraverso il **SessionDataHandler** direttamente dai metodi delle classi, in modo completamente trasparente per le classi che invocano i metodi).

L'uso di **HashMap** in **SessionDataHadler** garantisce l'accesso in tempo costante, in modo da non compromettere eccessivamente le prestazioni del sistema ma permettendo al contempo di mantenere solo parte degli oggetti in memoria.

2.2 Il resto delle classi appartenenti al package Controller sono emersi dall'analisi entity-boundary-controller

3. Dao

Ogni classe del DAO fornisce le query per l'accesso ai dati di un particolare sotto-sistema dell'applicazione, spesso per ogni controller corrisponde una classe del DAO.

I controller infatti si servono delle query messe a disposizione dai metodi delle classi per svolgere le loro operazioni sul database. I nomi dei metodi spesso contengono parole che richiamano le operazioni tipiche che possiamo svolgere sul database, per esempio:

- insert per i metodi che aggiungono nuovi record al db.

- load per i metodi con query che caricano record dal db.

- update per i metodi che aggiornano record nel db.

- delete per i metodi che cancellano record dal db.

Corrispondenze tra i sottosistemi e le classi del DAO:

VIEWER:

UserAuthenticationQuerySet per login e registrazione.

RequestToAdminQuerySet per richiedere ruoli all'amministratore o per altre esigenze.

EditProfileQuerySet per l'aggiornamento del proprio profilo utente.

HomePageQuerySet per caricare le informazioni principali da visualizzare nella homepage.

PageViewQuerySet per caricare le pagine di un Document da visualizzare nella PageView.

SearchQuerySet per la ricerca di Document all'interno del db.

UPLOADER:

DigitalizerQuerySet per caricare e aggiornare le scansioni di un'opera.

RevisorDigitalizationQuerySet per caricare gli esiti delle revisioni delle scansioni.

TRANSCRIBER:

TranscriberQuerySet per caricare e aggiornare le trascrizioni delle pagine.

TranscriptionRevisorQuerySet per caricare gli esiti delle revisioni delle trascrizioni.

MANAGER:

DocumentQuerySet per inserire nuove opere nel db.

ScanningWorkProjectQuerySet per inserire e aggiornare i progetti di digitalizzazione.

TranscriberWorkProjectQuerySet per inserire e aggiornare i progetti di trascrizione.

ADMINISTRATOR:

AdministratorQuerySet per inserire/rimuovere/aggiornare utenti e ruoli.

WORK SESSION HANDLER

WorkSessionQuerySet contiene in particolare due metodi startQuerySession(user, entities) ed endQuerySession(). Sostanzialmente ci sono delle entità nel database che possono essere modificate solo da un utente per volta, per esempio solo un trascrittore per volta può avere accesso alla trascrizione di una pagina, o solo un digitalizzatore per volta può avere accesso all'intero set di scansioni dell'opera, stesso discorso vale per i rispettivi revisori. Dunque abbiamo scelto di inserire dei *flag* all'interno del db per garantire la mutua esclusione per gli accessi. Il metodo startQuerySession(user, entities) restituisce true al controller di una certa attività se l'utente può accedere a tali entities, viceversa restituisce falso nel caso in cui la sessione è occupata già da un altro utente. Il controller comunicherà l'esito all'utente nella maniera più opportuna.