**PROPOSTA DI PROGETTO PER LA PROVA FINALE DEL CORSO DI**
**OBJECT ORIENTED SOFTWARE DESIGN**
**CORSO DI LAUREA IN INFORMATICA**
**UNIVERSITA' DEGLI STUDI DELL'AQUILA**
**A.A. 2017/2018**

## SPECIFICA

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 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.

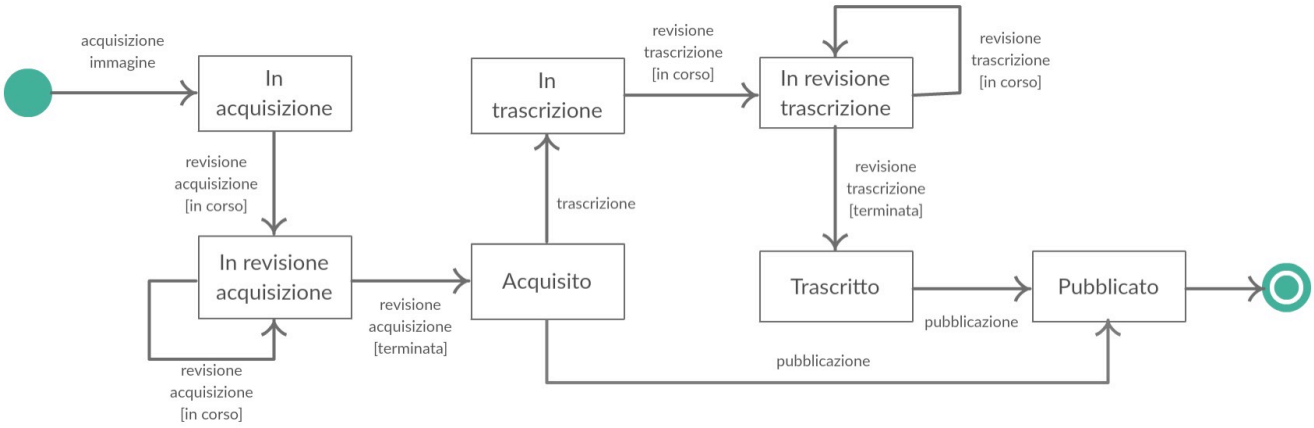
*Transcriber*
Ogni opera acquisita deve essere trasformata in un testo digitale, ciò avviene attraverso operazioni di trascrizioni in formato TEI (Text Encoding Initiative). Le trascrizioni sono digitate manualmente

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.

Processo di pubblicazione:




Links utili:

https://transkribus.eu/Transkribus/

**REQUISITI DEL PROGETTO:**

## Parte 1

1. **Requirements**
   Riorganizzare la specifica data sotto forma di requisiti.
   *Documenti richiesti: (1.1) documento dei requisiti, (1.2) modelli UML use case, (1.3) modelli di dominio (classi entity complete), (1.4) analisi finalizzata all'individuazione di classi entity, boundary e controller*

2. **System design**
   Definire gli obiettivi di design, decomporre il sistema, selezionare le strategie per la costruzione del sistema, definire l'architettura software del sistema, scegliere dei pattern architetturali/design (tra cui, MVC).
   *Documenti richiesti: (2.1) modello dell'architettura software (o più modelli UML con package con diverso grado di dettaglio), (2.2) descrizione dell'architettura, (2.3) descrizione delle scelte e strategie adottate (compresi pattern architetturali/design)*

3. **Software/Object design**
   Definire gli oggetti del dominio, definire classi, interfacce e membri.
   *Documenti richiesti: (3.1) UML class diagrams rappresentanti l'object design con classi, interfacce e membri, (3.2) descrizione dei dettagli di design scelti.*

4. **Deliverable (deadline 28/4/2018)**
   Il progetto deve essere mantenuto in un repository GIT, che sarà strutturato come segue.

   *<nome_progetto>* : root del progetto
   - *src :* questa directory conterrà il codice sorgente del progetto
   - *doc :* questa directory conterrà i documenti di design del progetto (in format pdf)
   - *javadoc :* questa directory conterrà il codice javadoc del progetto

   La URL del repository sarà consegnata al docente, la parte di competenza di questo deliverable sarà contenuta nella directory doc.

*Note:* Gli strumenti a disposizione per svolgere la parte 1 del progetto riguardano il programma del corso relativo alle lezioni 1-13 (Object-Oriented Software Engineering e Java Foundations).

## Parte 2

1. **Refining the design**
   Il design del sistema e il design software devono essere raffinati ed integrati alla base delle nuove conoscenze acquisite nella seconda parte del corso.
   *Documenti da raffinare: (2.1), (2.2), (2.3), (3.1)*

2. **Implementation**
   Implementazione del sistema software alla base delle scelte di design effettuate. Il lavoro deve essere svolto all'interno dell'ambiente di sviluppo Eclipse. Il lavoro deve essere sviluppato in modo collaborativo attraverso un repository GIT.

   L'implementazione del sistema deve necessariamente far uso dei seguenti strumenti:
   - Classi, package e modificatori di accesso per una corretta strutturazione del sistema

- Interfacce
- Ereditarietà
- Polimorfismo e overriding/overloading di metodi con almeno un esempio di dinamic binding
- Uso di final, static
- Eccezioni
- Collection/Map
- JDBC

3. **Documentation**
   La documentazione da allegare al progetto sarà costituita dai documenti ai punti 2.* e 3.* raffinati e inoltre da Javadoc ed eventuali annotazioni (ciò implica opportuna formattazione e commenti).

4. **Deliverable**

   A ultimazione del progetto la comunicazione della URL del repository da aggiorare sarà consegnata al docente, le parti di competenza di questo deliverable saranno contentute nelle directory src, doc, javadoc.

*Note:* Gli strumenti a disposizione per svolgere la parte 2 del progetto riguardano il tutto programma del corso.

**Tutti i progetti non conformi ai sopra indicati requisiti non saranno accettati.**