“Object oriented programming”
a.a. 2015-2016

Biblioteca virtuale

Team Members

Name & Surname: Leonardo Ruggeri

Matriculation: *231254*

E-mail address: leonardo.ruggeri@student.univaq.it

Indice

A. Requirements Collection

Che cosa deve fare il sistema?

Prima di passare all'analisi dei requisiti descriviamo brevemente cosa deve fare il sistema.

Il sistema deve gestire un insieme di **collezioni, servizi e persone** in modo da supportare le fasi di **creazione, uso, preservazione dei dati e diffusione.**
Deve essere in grado, tramite un'interfaccia grafica, di permettere ad un utente di immagazzinare immagini.
Successivamente il testo dovrà essere trasformato in un formato digitale attraverso un text editor TEI integrato.
Il sistema deve consentire a degli utenti speciali di supervisionare l'acquisizione delle immagini e la traduzione in formato digitale.
Le opere deve poter essere pubblicate e successivamente consultate dagli utenti.

A.1 Functional Requirements

Sono stati identificati i seguenti *attori* del sistema:

- **Amministratore**:
  Gestione generale del sistema
- **Acquisitore**:
  Acquisizione/digitalizzazione immagine
- **Revisore acquisizioni**:
  Revisione e verifica correttezza dell'acquisizione
- **Trascrittore**:
  Trascrizione del testo TEI
- **Revisore trascrizioni**:
  Revisione e validazione della trascrizione
- **Utente base**:
  Visualizzazione elenco titoli opere
- **Utente avanzato**:
  Visualizzazione completa delle opere

Priorità dei requisiti

1. Trascrizione tramite un text editor TEI
2. Controllo delle immagini
3. Controllo del testo
4. Immissione immagini
5. Pubblicazione immagini e testo
6. Consultazione titoli
7. Consultazione contenuti

I requisiti di maggior importanza sono la trascrizione del testo e il suo controllo in quanto lo scopo principale del sistema è la digitalizzazione di manoscritti.
Viene data minor importanza alla consultazione dei titoli e dei contenuti in quanto l'operazione all'inizio verrà poco eseguita.

A1.1 Requisiti di digitalizzazione

Immissione immagini

Il manoscritto è acquisito dal sistema sotto forma di immagini digitali ad alta risoluzione attraverso scanner planetari. Ogni manoscritto è formato da più acquisizioni (ogni immagine rappresenta una singola pagina).
L'immagine acquisita viene memorizzata all'interno del sistema ed assegnata all'opera di riferimento, corredandola di opportuni metadati.

Controllo immagini

La digitalizzazione viene controllata da supervisori all'acquisizione per assicurarne la correttezza (ad esempio, in accordo con standard richiesti) e la qualità.

A1.2 Requisiti di trascrizione

Trascrizione tramite un text editor TEI

Il manoscritto così acquisito deve essere trasformato in un testo digitale; ciò avviene attraverso operazioni di trascrizioni in formato TEI (Text Encoding Initiative)[1]. Le trascrizioni sono digitate manualmente (la natura del testo rende inutilizzabili strumenti di acquisizione automatica) attraverso un text editor TEI integrato[2].

Controllo testo

Le trascrizioni sono oggetto di revisione da parte di revisori alle trascrizioni.

A1.3 Requisiti di pubblicazione

Pubblicazione immagini e testo

I manoscritti, una volta digitalizzati e superata la fase di revisione delle immagini, vengono pubblicati sul sistema e resi accessibili agli utenti del sistema. Le corrispondenti trascrizioni sono pubblicate successivamente, dopo la validazione della stessa.

Consultazione contenuti

L'utente deve poter consultare i titoli e le opere

A.2 Non Functional Requirements

Usability: il Sistema deve poter essere utilizzato facilmente da qualsiasi tipo di utente

B. Analysis Model

Dall' analisi dei requisiti si individuano le seguenti classi, suddivise nelle tre tipologie previste da UML:

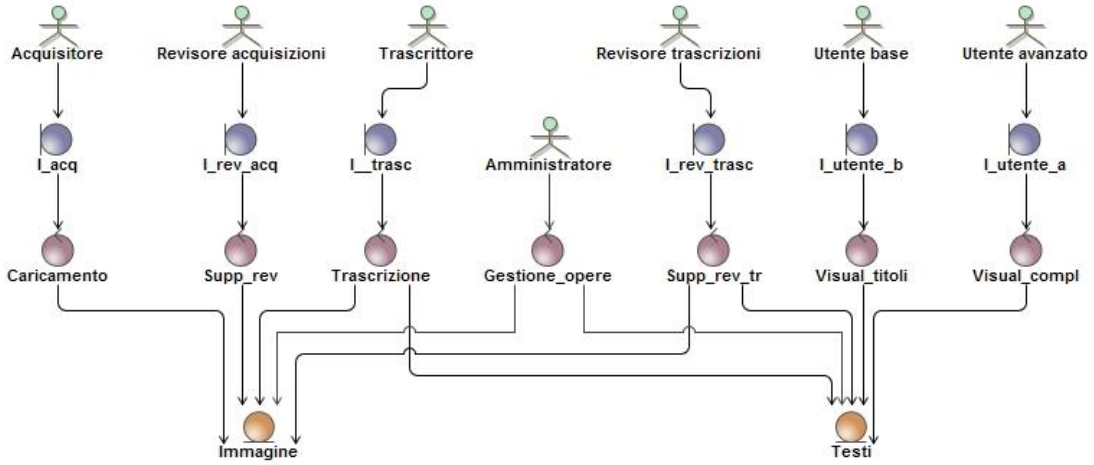


Figura 1: Robustness Diagram

Classi Boundary: rappresentano l'interfaccia che consente agli utenti di interagire col sistema, nel nostro caso gli utenti utilizzano molte interfacce grafiche, questo perché ogni utente deve svolgere una diversa operazione. Questo ci permetterà in seguito di poter gestire separatamente le diverse interfacce.

Classi Entity: I dati che il sistema andrà a manipolare sono di tipo "immagine" e "testo",
E' quindi sufficiente prevedere due classi per rappresentare i dati del sistema.

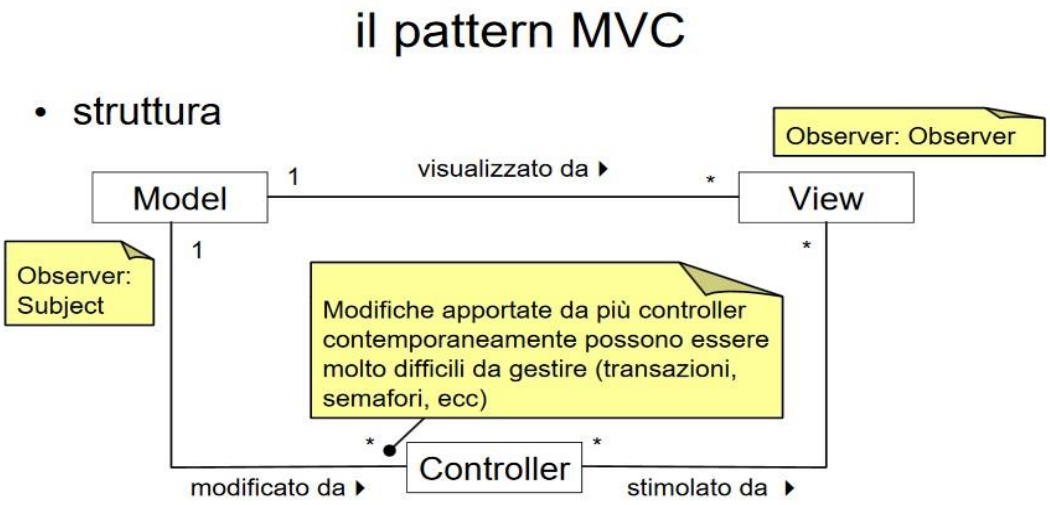
Classi Controller: Conterranno le funzioni di manipolazione degli oggetti "testo" e "immagine" che prendono in ingresso parametri ricevuti dalle classi "Boundary".

Queste funzionalità rappresentano ad alto livello cosa il sistema dovrà fare .

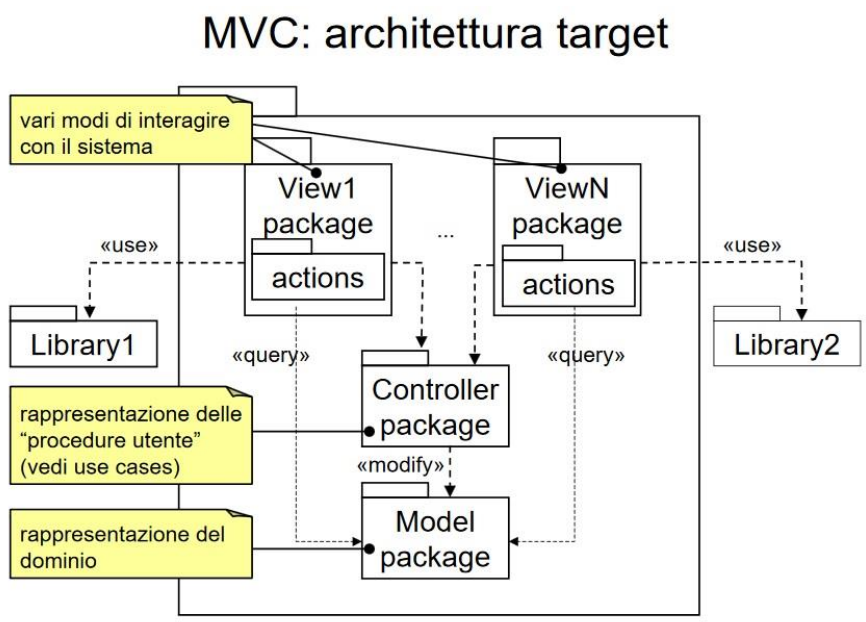
C. Software Architecture

Come modello architetturale è stato scelto il pattern MVC per le sue peculiarità nella separazione delle logiche, infatti un sistema è gestibile se è diviso in parti che siano indipendenti tra loro.
Il pattern architetturale, nell'intento di disaccoppiare il sistema, stabilisce tre parti in cui esso verrà diviso:

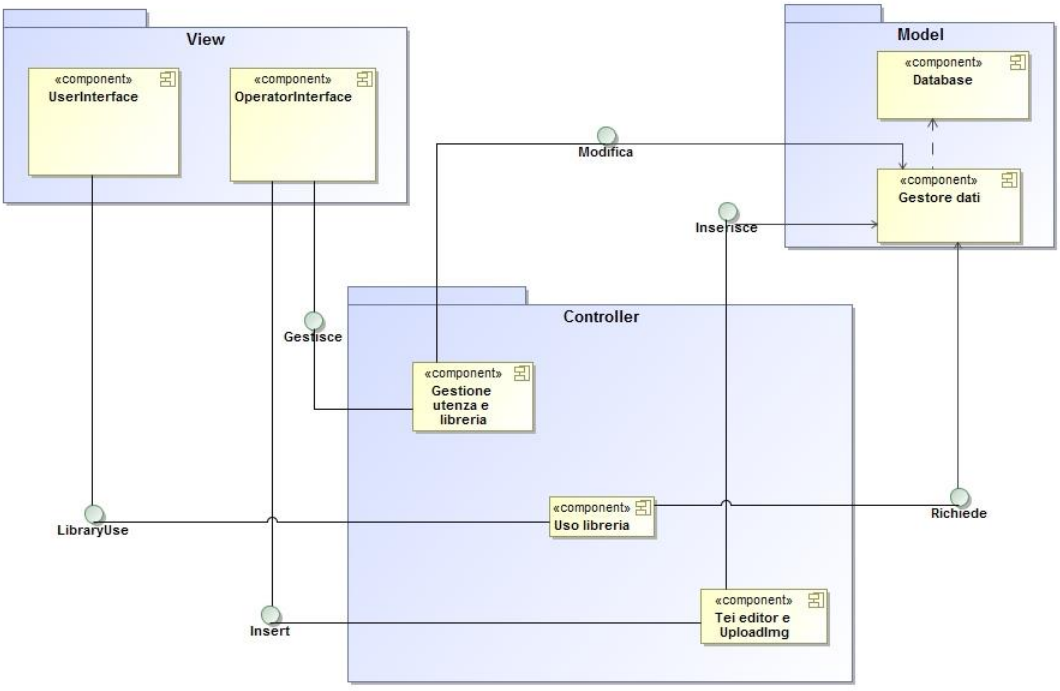
- rappresentazione del modello di dominio (model)
- interfaccia utente (view),
- controllo dell'interazione uomo-macchina (controller)



Dal modello architetturale teorico nell'immagine in alto possiamo estrapolare una rappresentazione del seguente tipo:


MVC: architettura target
vari modi di interagire con il sistema
View1 package
actions
ViewN package
actions
...
«use»
«use»
Library1
Library2
«query»
«query»
Controller package
rappresentazione delle "procedure utente" (vedi use cases)
«modify»
Model package
rappresentazione del dominio

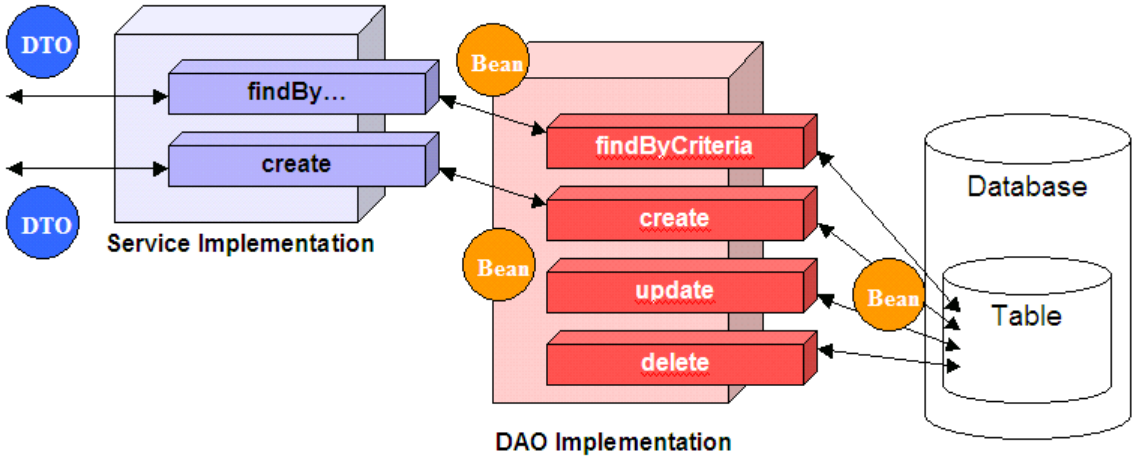
E quindi:


View
«component» UserInterface
«component» OperatorInterface
Model
«component» Database
«component» Gestore dati
Modifica
Inserisce
Gestisce
Controller
«component» Gestione utenza e libreria
«component» Uso libreria
«component» Tei editor e UploadImg
LibraryUse
Insert
Richiede

Inoltre per la gestione persistente dei dati viene sfruttato il design pattern DAO, cioè degli oggetti di accesso ai dati che gestiscono l'accesso a questi ultimi, incapsulando le modalità di comunicazione con il DB.

Le Data Access Objects si fanno carico di gestire il codice SQL, mentre ciò è trasparente rispetto alle corrispondenti classi di dominio e di controllo.

A livello di logica dell'applicazione siamo fortemente orientati agli oggetti, infatti ragioniamo solo in termini di Domain Objects, cioè dei concetti pertinenti al dominio dell'applicazione, e non possiamo mai utilizzare i metodi di accesso diretto alla base dati forniti dai DAO.
Il design pattern DAO quindi ci dà un approccio diverso alle future modifiche e manutenzioni poiché i dati vengono gestiti da apposite classi, separando cosi in modo migliore le logiche a vantaggio di chi dovrà gestire il codice in futuro.



In generale, lo scopo principale di un Component Diagram è di mostrare le relazioni strutturali tra le componenti di un sistema (una componente è un elemento fisico rimpiazzabile del sistema).
Le componenti coinvolte sono le seguenti:

OperatorInterface, UserInterface, Gestione utenza e libreria, Tei editor e Uploading, Uso libreria, Database.

1. ***UserInterface***: Rappresenta le interfacce grafiche con le quali l'utente utilizza il sistema. Comunicherà con "Uso libreria" attraverso l'interfaccia "library" passandogli i valori per una eventuale ricerca.

2. ***OperatorInterface***: Rappresenta le interfacce grafiche con le quali l'operatore utilizza il sistema.
   Si Comunicherà con "Gestione utenza e libreria" attraverso l'interfaccia "Gestisce".
   Operator interface è una view comune a Amministratore e Validatori, questo per accorpare le interfacce gestionali.

3. ***Database***: Rappresenta la parte Model del sistema: il database JDBC, il quale si occuperà di contenere i dati.

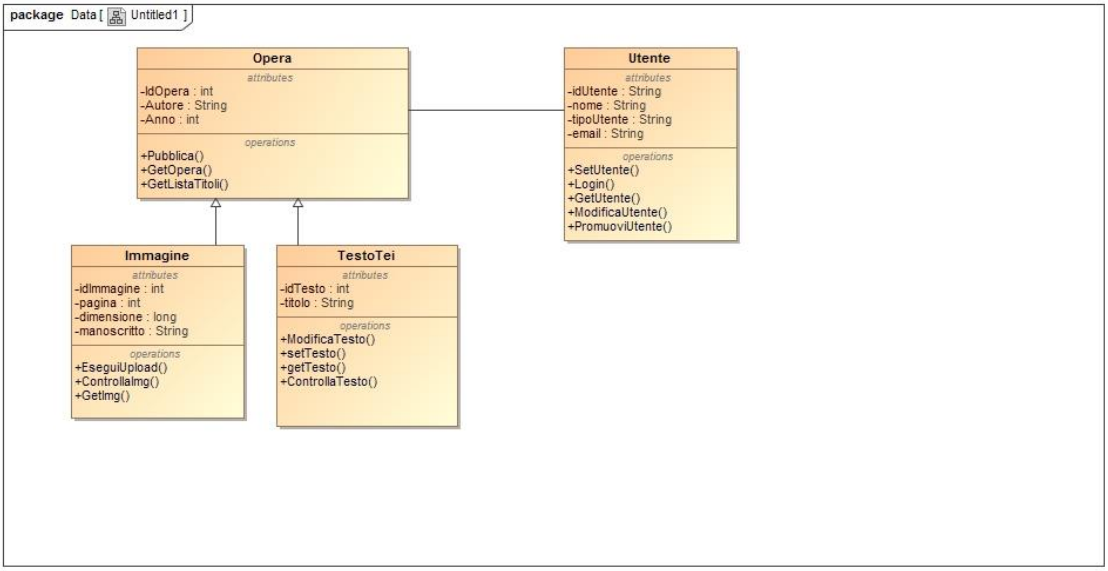
4. ***Gestione dati***: Costituisce la parte del sistema che si occupa di estrapolare i dati dalla base dati

5. ***Gestione utenza e libreria***: La parte controller del sistema dove saranno accorpate tutte le procedure funzionali.

6. ***Uso libreria***: Raggruppa tutte le funzionalità per utilizzare la libreria.

7. ***Tei editor e UploadImg***: Rappresenta le funzionalità di scrittura di un testo TEI e di upload di un'immagine. Sono state accorpate perché definiscono la parte back-end del sistema in quanto generano i contenuti del sistema.

D. Class Diagram of the implemented System



Il class diagram consente di descrivere *tipi di entità*, con le loro caratteristiche e le eventuali relazioni fra questi tipi.
Per il nostro sistema sono state individuate le seguenti classi:

- La classe **Utente** che racchiude i dati di tutti gli utenti e permette di fare operazioni su di essi.

- La classe **Opera** permette di gestire e ottenere le opere, ha due sottoclassi: **Immagine** e **TestoTei** che rappresentano le immagini e i testi nel formato TEI, e contengono le funzionalità per gestirli

E. Design Decisions

Tra le scelte architetturali più importanti rientra quella di creare un'applicazione desktop o una applicazione web; uno dei vantaggi di un'applicazione desktop è sicuramente la possibilità di essere utilizzata in assenza di rete e di mantenere file in locale, quest' ultima peculiarità porterebbe ad un minor sovraccarico del server.

Tuttavia, il problema di utilizzare i file in assenza di rete, può essere superato fornendo all'utente la possibilità di scaricare le informazioni da lui ricercate in un qualunque formato.

È ritenuta migliore la scelta architetturale di utilizzare una applicazione web per la sua facilità di utilizzo per gli tutti gli utenti e la possibilità di accedere al sistema da qualunque piattaforma.

Per la creazione del Sistema è stato scelto di utilizzare un'architettura web-oriented, che si presta alla gestione della multiutenza, il sistema infatti prevede che ci siano sette diversi tipi di utenti e che ognuno abbia diversi scopi.

Per la gestione della multiutenza si prevede che il database sia creato in modo da avere tre diverse tabelle che gestiscano:

- Utenti
- Gruppi di utenti
- Servizi

Nella tabella utenti ci saranno i dati di ciascun utente, nella tabella gruppi di utenti ci saranno i sette diversi tipi di utenti e nella tabella servizi le funzionalità a cui possono accedere i gruppi di utenti.

Gli amministratori del sistema si occuperanno di assegnare un utente ad un particolare gruppo di utenti, ad eccezione per l'utente avanzato che viene assegnato al suo gruppo automaticamente dopo la registrazione.

L'applicazione web dovrà fornire a ciascun tipo di utente la sua interfaccia, in modo da permettergli di svolgere la sua mansione.