DIETI UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Sistema di gestione di una pagina wiki

Progetto basi && object

Piervincenzo Salierno
Davide Spasiano

November 2023

Indice

1 Descrizione di progettazione del sistema

Lo scopo di questa fase è verificare le informazioni fondamentali e le istruzioni di base per definire la struttura di un sistema per la gestione di una pagina WIKI.
Quindi sarà effettuata un'analisi che sarà documentata per blocchi di requisiti, si riportano di seguito i primi requisiti individuati nel documento fornito come input al progetto:

Si sviluppi un sistema informativo per la gestione del ciclo di vita di una pagina wiki.

R1: Ogni pagina wiki ha un titolo e un testo.

R2: Ogni pagina è creata da un determinato autore.

R3: Il testo è composto di una sequenza di frasi.

R4: Il sistema mantiene traccia anche del giorno e ora nel quale la pagina è stata creata.

Considerando la richiesta di sviluppare un sistema per la gestione di pagine WIKI, viene naturalmente da pensare a distinguere ogni pagina da un'altra. Quindi per rappresentare ogni pagina del sistema, verrà utilizzata una classe **PAGINA** caratterizzata dagli attributi *Titolo* e *DataOra*
Un ulteriore requisito è quello di mantenere nel sistema il testo di ogni pagina definito come una lista ordinata di frasi. Per questo motivo è preferibile creare direttamente una classe **FRASE** per prendere singolarmente ogni frase. In questo modo si andrebbe anche ad allegerire il database, visto che non ci sarà bisogno di creare un ulteriore tabella per il testo e operare direttamente sulle singole tuple della tabella **FRASE**.
Di conseguenza ci sarà una relazione tra **PAGINA** e **FRASE** di tipo 1 a N. Tra gli attributi principali abbiamo sicuramente *parola*, *DataOra* e *OrdineFrase* che definisce la posizione della frase all'interno della pagina.

Continuando l'analisi con un nuovo blocco di requisiti identificati nel documento iniziale:

R5: La pagina può contenere anche dei collegamenti.

R6: Ogni collegamento è caratterizzato da una frase da cui scaturisce il collegamento e da un'altra pagina destinazione del collegamento.

R7: Il testo può essere modificato da un altro utente del sistema, che seleziona una o più delle frasi, scrive la sua versione alternativa (modifica) e prova a proporla.

R8: Nel caso in cui si fossero accumulate più modifiche da rivedere, l'autore dovrà accettarle o rifiutarle tutte nell'ordine in ordine dalla più antica alla più recente.

Per gestire eventuali collegamenti tra pagine è stata aggiunta un'ulteriore relazione tra pagina e frase, anche in questo caso è di tipo 1 a N. Di conseguenza una tupla di frase avrà due riferimenti di pagina, una che si riferisce alla pagina di appartenenza e una che si riferisce alla pagina con cui può avere un collegamento.
Il sistema deve prevedere 2 tipologie di utenti registrati nel sistema: gli utenti, chiamati autori, che hanno la possibilità di creare pagine o modificare pagine di altri utenti e gli utenti che possono leggere le pagine pubblicate nel sistema.

E' stata quindi utilizzata una superclasse **UTENTE**, ed una *specializzazione* con la sottoclasse **Autore** di tipo *Totale,disgiunta*.
Ogni **UTENTE** sarà identificato dal *Codice Fiscale* che sarà inevitabilmente univoco per ogni utente e per essere valido dovrà avere 9 caratteri alfa-numerici, altrimenti non verrà accettato dal sistema. Inoltre la classe avrà come attributi i principali campi per la registrazione di un utente (*nome,cognome,mail,password*).
Gli utenti generici del sistema (cioè coloro che non hanno eseguito l'accesso o non hanno un account) non hanno bisogno di essere registrati nel sistema, ma potranno comunque accedere in sola lettura andando a cercare e visualizzare le pagine della WIKI.

Continuando l'analisi con l'ultimo blocco di requisiti identificati nel documento iniziale:

R9: La modifica proposta verrà notificata all'autore del testo originale la prossima volta che utilizzerà il sistema.

R10: L'autore potrà vedere la sua versione originale e la modifica proposta. Egli potrà accettare la modifica (in quel caso la pagina originale diventerà ora quella con la modifica apportata), rifiutare la modifica (la pagina originale rimarrà invariata). La modifica proposta dall'autore verrà memorizzata nel sistema e diventerà subito parte della versione corrente del testo.

R11: Il sistema mantiene memoria delle modifiche proposte e anche delle decisioni dell'autore (accettazione o rifiuto).

Un altro requisito fondamentale è gestire le modifiche da parte di altri utenti o magari da parte dell'utente stesso della pagina, mantenendo i record all'interno del database. Per fare ciò è necessario introdurre un attributo *type* che andremo a chiamare *stato* e indicherà lo stato di quella frase nel sistema. I possibili valori che potrà assumere quest'attributo sono: *accettato, sostituito, rifiutato, in_attesa*.

***Accettato*:** indica che la frase è ammessa nell'ultima versione della pagina.

***Rifiutato*:** indica la tupla inerente a una modifica che l'autore di quella pagina ha rifiutato da parte di un altro utente del sistema.

***Sostituito*:** indica la tupla che è stata sostituita da una nuova tupla.

***in_attesa*:** indica la tupla di una modifica proposta da un utente e che deve essere accettata o rifiutata dall'autore di quella pagina.

E' presente anche un attributo *versione* che identifica la versione di quella frase, che viene incrementata ogni volta che viene inserita una nuova modifica alla frase. Per esempio, quando viene inserita per la prima volta una frase la versione assume il valore di default 1, se viene proposta una modifica a quella frase, la nuova frase inserita avrà versione 2 e via dicendo. Inoltre è presente anche una relazione ricorsiva in modo da garantire una coerenza delle modifiche ai testi delle pagine. Questa relazione è del tipo 1 a N. Per esempio: a una frase con stato "accettato" e quindi visibile nella pagina, vengono proposte 2 modifiche da parte di altri utenti. Quindi vengono inserite 2 nuove tuple nel database ed entrambe faranno riferimento alla stessa tupla. Mentre ogni frase inserita può avere 0 o una sola frase a cui fare riferimento.
E' presente inoltre un attributo *DataOraApprovata* per mantenere l'ora esatta in cui una modifica viene accettata nel sistema.

Nel corso dell'analisi è emerso che il sistema deve prevedere una componente per l'interazione con gli utenti (gui) e di una componente per la persistenza dei dati gestiti dal sistema (database).
Al fine di rispettare i principi:

- Leggerezza della GUI
- Separazione tra i package
- Comunicazione tra le finestre

è stato deciso di usare come architettura del sistema il modello BCED.
Nei paragrafi successivi sono descritti i package Model, Controller, GUI e DAO. E' inoltre riportato il diagramma di dettaglio delle classi del sistema e il sequence diagram per due funzionalità del sistema.

1.1 Diagramma delle classi del dominio del problema

Le classi identificate precedentemente rappresentano le entità del sistema, sono quindi presenti all'interno del package **model**. Di seguito se ne riporta il diagramma UML.

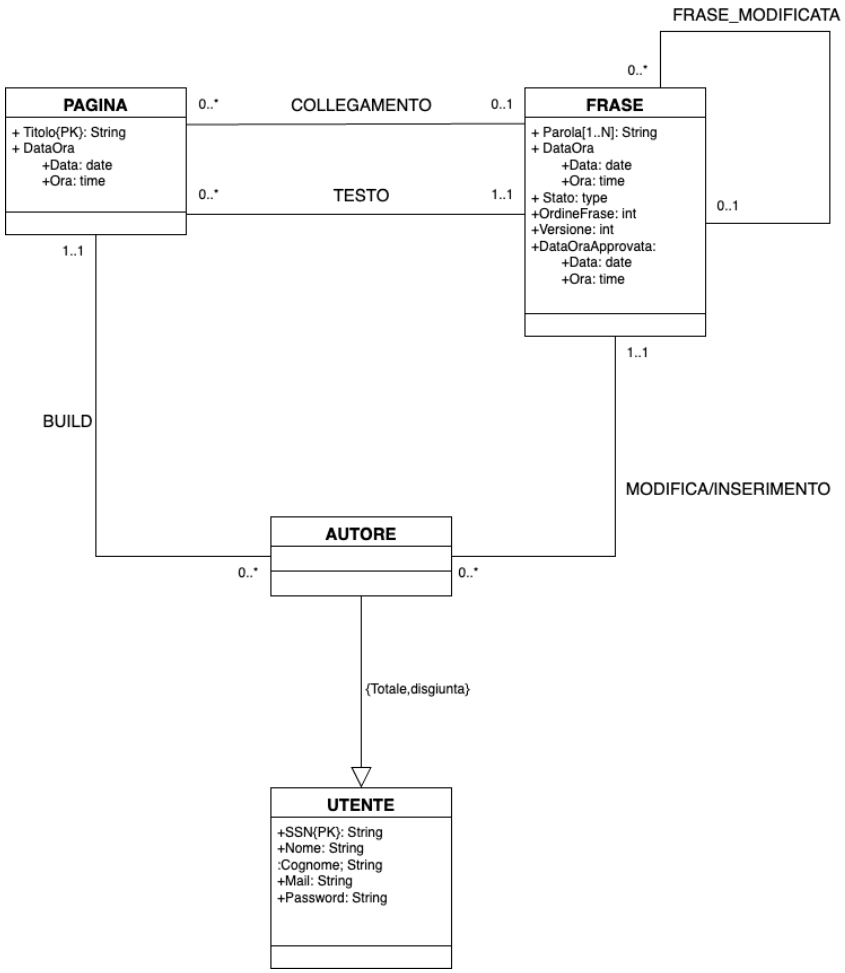


Figure 1: Modellazione UML

1.2 Controller

Le classi del package **model** sono collegate a una classe **Controller** presente all'interno del package **controller**. Il controller realizza tutte le operazioni algoritmiche su richiesta diretta della GUI e gestisce i dati rappresentati nel model. Quindi le funzionalità del controller sono:

- è responsabile delle funzionalità del sistema
- riceve richieste solo dalla GUI
- chiede letture e scritture dei dati conservati nel model
- è unico e non viene distrutto prima della chiusura del programma

1.3 Database e DAO

In questo sistema è fondamentale avere un database all'interno del quale andranno mantenuti i dati in modo persistente. Quindi si dovrebbe creare una classe Database chiamata da ogni classe del model, con una interazione diretta con il database. Da un punto di vista di architettura, questa è una soluzione a livelli perfettamente separati e la progettazione del controller può avvenire senza sapere quale database verrà utilizzato e nemmeno se verrà utilizzato o meno un database per mantenere i dati persistenti. Tuttavia il principale difetto è che il Model progettato inizialmente mantiene dipendenze dallo specifico database e deve essere modificato per supportare database differenti.

Per questo motivo è preferibile utilizzare il pattern **DAO** (Data Access Object).
Quindi si introdurrà un package **dao** che contiene solo *interface*, relativamente a tutti i metodi che andranno realizzati tramite query sul database per gestire la persistenza dei dati.
In questo caso abbiamo 3 interface: **PaginaDAO, FraseDAO, UtenteDAO**.

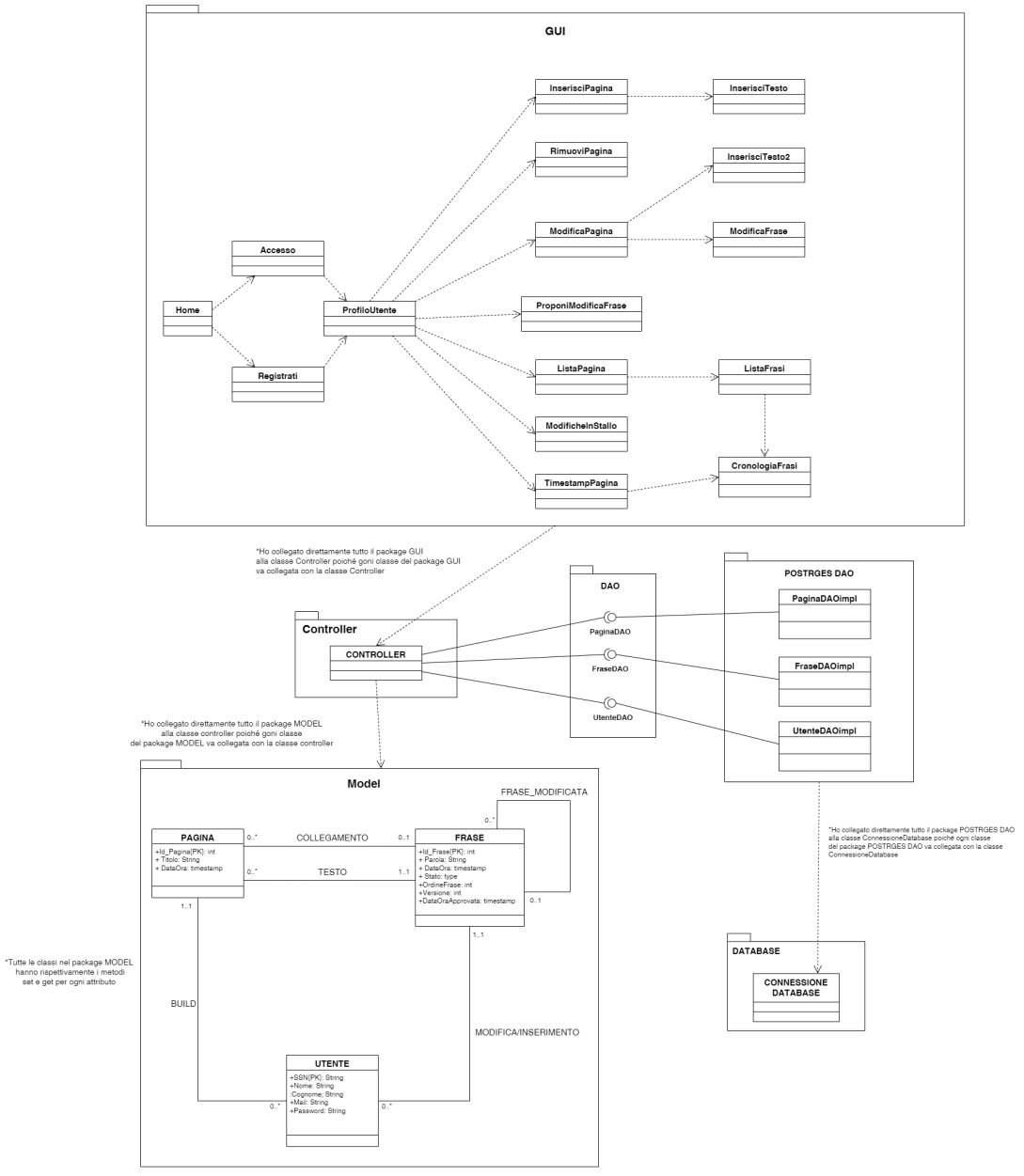
Per ogni database supportato verrà introdotto un package **implementazioneDAO** (in questo caso specifico utilizzando Postgres, il package è **implPostgresDAO**).
Questo package implementa con classi concrete (**PaginaDAOimpl**, **FraseDAOimpl**, **UtenteDAOimpl**) le interfacce del package dao, realizzando le query necessarie per il database.
La connessione al database invece viene gestita da una classe di utilità **ConnessioneDatabase**.

1.4 GUI

La GUI utilizzata per l'iterazione degli utenti del sistema è **Java Swing**, che permette una realizzazione di un'interfaccia semplice e intuitiva ma al tempo stesso complessa e ricca di funzionalità.
L'elemento basilare di una GUI è il **form**, che coincide con una finestra dell'applicazione.
All'avvio, il main form è **Home**, e permette l'accesso all'applicazione a chiunque, quindi anche agli utenti generici del sistema i quali possono eseguire delle ricerche sulle pagine esistenti nella WIKI.
Dalla Home si può passare all'accesso o alla registrazione di un utente, tramite due appositi bottoni, i quali portano rispettivamente alle form **Accesso** e **Registrazione**.

Da qui una volta inseriti tutti i dati necessari, si potrà tornare indietro alla Home, oppure avanti alla form **Profilo utente** che rappresenta appunto tutte le funzionalità e le operazioni che può eseguire un utente del sistema, ad esempio: creazione/rimozione/modifica pagina, gestione collegamenti, proposta di modifica, modifiche in stallo, cronologia versioni pagina e cronologia versioni frase, eliminazione account e infine logout che ti porta al form principale.

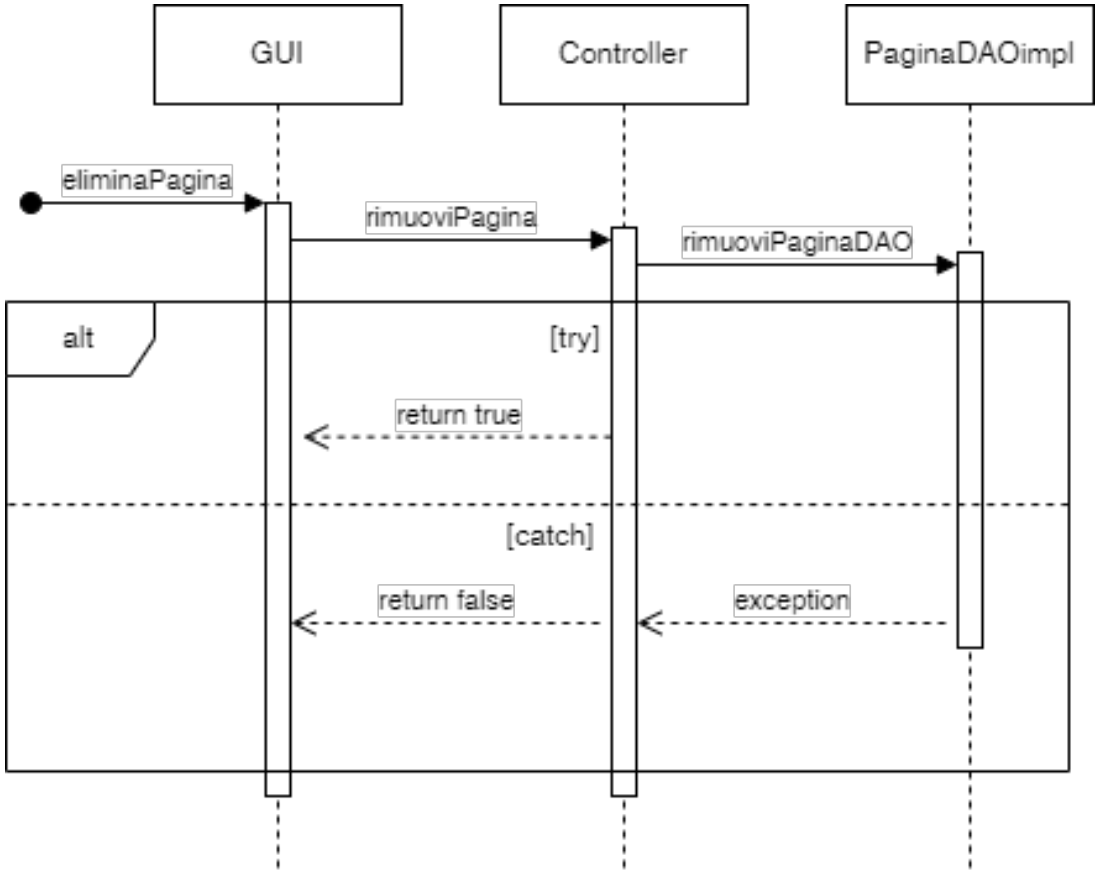
1.5 Diagramma di dettaglio della classi nel dominio della soluzione



1.6 Sequence Diagram

Il sequence diagram è il diagramma di interazione più largamente utilizzato ed è molto utile per modellare le interazioni tra uno o più attori e il sistema software, nell'ambito dell'esecuzione di uno scenario di esecuzione. In altre parole sono utili per far comprendere i vari passaggi all'avvenire di uno specifico evento.

Il primo sequence diagram rappresenta la rimozione di una pagina dal sistema.



Il secondo sequence diagram rappresenta l'accesso al sistema da parte di un utente.