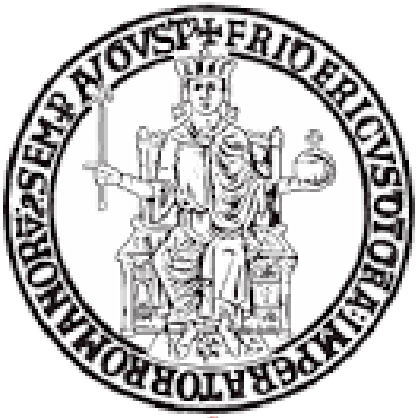


UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II

Progetto Object-Orientation

Creazione di un'applicazione che si occupa della gestione di pagine wiki

Anno accademico 2023/2024

Corso informatica

Autori:
Mirko Prevenzano N86004082
Giuseppe Patalano N86004118

Indice

1 Dominio del problema

1.1 Traccia

Si sviluppi un sistema informativo, composto da una base di dati relazionale e da un applicativo Java dotato di GUI (Swing o JavaFX), per la gestione del ciclo di vita di una pagina di una wiki. Una pagina di una wiki ha un titolo e un testo. Ogni pagina è creata da un determinato autore. Il testo è composto di una sequenza di frasi. Il sistema mantiene traccia anche del giorno e ora nel quale la pagina è stata creata. La pagina può contenere anche dei collegamenti. Ogni collegamento è caratterizzato da una frase da cui scaturisce il collegamento e da un'altra pagina destinazione del collegamento. 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. La modifica proposta verrà notificata all'autore del testo originale la prossima volta che utilizzerà il sistema. 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. Il sistema mantiene memoria delle modifiche proposte e anche delle decisioni dell'autore (accettazione o rifiuto). 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. Ad esempio, il testo originale del 3 novembre ore 10 del Prof. Tramontana potrebbe essere "traccia del progetto di OO", mentre il 3 novembre alle ore 11 il Prof. Barra potrà modificarlo in "traccia del progetto di OO e BD" e il Prof. Tramontana potrà accettare la modifica alle ore 12. A quel punto la versione corrente sarà quella proposta dal Prof. Barra. In alternativa il Prof. Tramontana potrebbe, alle ore 12 rifiutare la modifica del Prof. Barra e attuare invece la modifica "traccia del progetto di OO e BD gruppo 2" che diventerà subito parte della versione corrente (essendo la modifica stata disposta dall'autore della pagina). Gli utenti generici del sistema potranno cercare una pagina e il sistema mostrerà la versione corrente del testo e i collegamenti. Gli autori dovranno prima autenticarsi fornendo la propria login e password. Tutti gli autori potranno vedere tutta la storia di tutti i testi dei quali sono autori e di tutti quelli nei quali hanno proposto una modifica.

1.2 Analisi dei requisiti

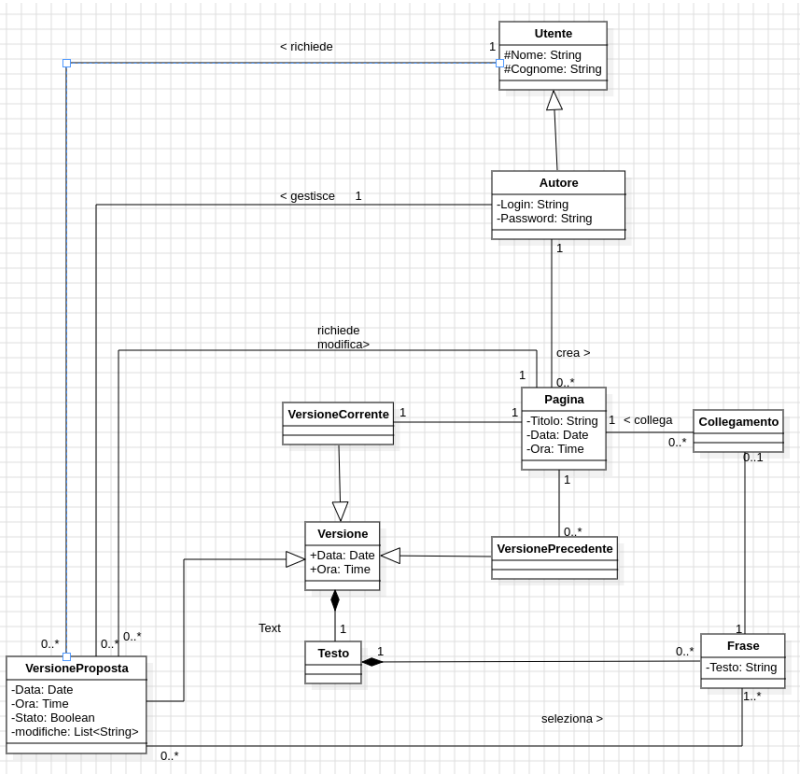
Il progetto richiede lo sviluppo di un'applicazione Java per la gestione di un sistema di pagine wiki. La classe centrale dell'applicazione è la classe **Pagina**, da cui è possibile ottenere il titolo della pagina, la data e l'ora di creazione.

L'obiettivo principale è gestire il ciclo di vita di una pagina wiki. Ogni pagina ha una versione corrente e può ricevere proposte di modifica. Se accettate, queste proposte diventano versioni precedenti della pagina. Per implementare questo concetto, si è scelto di introdurre la classe **Proposta**, che generalizza **VersioneCorrente**, **VersionePrecedente** e **VersioneProposta**.

Ciascuna versione è composta da un testo, il quale è rappresentato dalla classe **Testo**, composta a sua volta da frasi. È richiesto che una frase possa contenere un collegamento verso un'altra pagina del testo, quindi è necessario definire la classe **Collegamento**. Ogni pagina è creata da un autore, mentre utenti generici possono effettuare azioni come per esempio proporre una modifica; per tale motivo si è scelto di implementare una classe **Utente** e una sua specializzazione **Autore**, in particolare un autore per accedere necessita di un username e una password, mentre l'utente solo nome e cognome.

1.3 Modello del dominio del problema

Qui è mostrato l'UML ricavato dalla fase precedente, in cui sono stati ricavati dalla traccia, i requisiti necessari



2 Struttura del Progetto con Modello BCE e Pattern DAO

Per questo progetto, è stata adottata una struttura basata sul modello BCE con il pattern DAO per l'implementazione del database.

2.1 Boundary (B)

Il Boundary rappresenta l'interfaccia che interagisce direttamente con l'utente. Le classi GUI, pertanto, non accedono direttamente alle informazioni del modello e non si occupano dell'elaborazione dei dati. Questo ruolo è svolto dal Boundary che si interfaccia con il Controller per gestire le operazioni tra l'utente e il sistema.

2.2 Controller (C)

Il Controller funge da tramite tra il modello e l'interfaccia utente. È responsabile di gestire tutti i dati e le operazioni da eseguire nel sistema. Il Controller è unico e riceve richieste solo dall'interfaccia utente. Si occupa di elaborare le richieste ricevute e di interagire con le classi del modello.

2.3 Entity (E)

Le Entità corrispondono alle classi del modello che rappresentano gli oggetti di dominio del sistema. Ricevono richieste solo ed esclusivamente dal Controller e non interagiscono direttamente con l'interfaccia utente.

Per quanto riguarda il database, per ogni entità presente nel database è stata implementata un'interfaccia, la quale è poi stata concretizzata da una classe apposita. Questa classe contiene i metodi necessari per l'accesso e la manipolazione dei dati relativi all'entità corrispondente nel database.

3 Traduzione del Modello del Dominio del Problema in Codice Java

Quando si traduce il modello del dominio del problema in codice, ogni classe nell'UML diventa una classe nel codice Java, rispettando le relazioni tra di loro.

3.1 Utente

L'entità Utente contiene gli attributi nome e cognome. Poiché un utente può proporre modifiche a determinate pagine, ha un riferimento a VersioneProposta, che implica la presenza di una lista di versioni.

3.1.1 Autore

Autore è una specializzazione di Utente con compiti specifici come la creazione di pagine e la gestione di proposte. È caratterizzato dalla presenza di una lista di pagine create e una lista di proposte da gestire. Per accedere al profilo, l'autore deve autenticarsi con username e password.

3.2 Pagina

La classe Pagina è centrale. È creata da un autore, pertanto contiene un riferimento ad esso. Ogni pagina ha una versione corrente, e può avere versioni proposte e versioni precedenti, quindi sono presenti due liste, una per le versioni precedenti e una per le versioni correnti.

3.3 Versione

Ogni versione è composta da un testo e include informazioni sulla data e l'ora di creazione. Versione è la generalizzazione di VersionePrecedente, VersioneCorrente e VersioneProposta.

3.3.1 VersioneCorrente

Rappresenta l'ultima modifica effettuata su una pagina e ha un riferimento ad essa.

3.3.2 VersioneProposta

Un utente può proporre modifiche a una pagina, quindi è riferito alle classi Utente e Pagina. La gestione della proposta avviene attraverso gli attributi elaborato e stato, che identificano lo stato della proposta. Un utente, nella modifica, seleziona una o più frasi e le modifiche proposte per ognuna di esse.

3.3.3 VersionePrecedente

Rappresenta le versioni precedenti di una pagina.

3.4 Testo

Ogni versione è composta da un testo, che consiste in una sequenza di frasi.

3.5 Frase

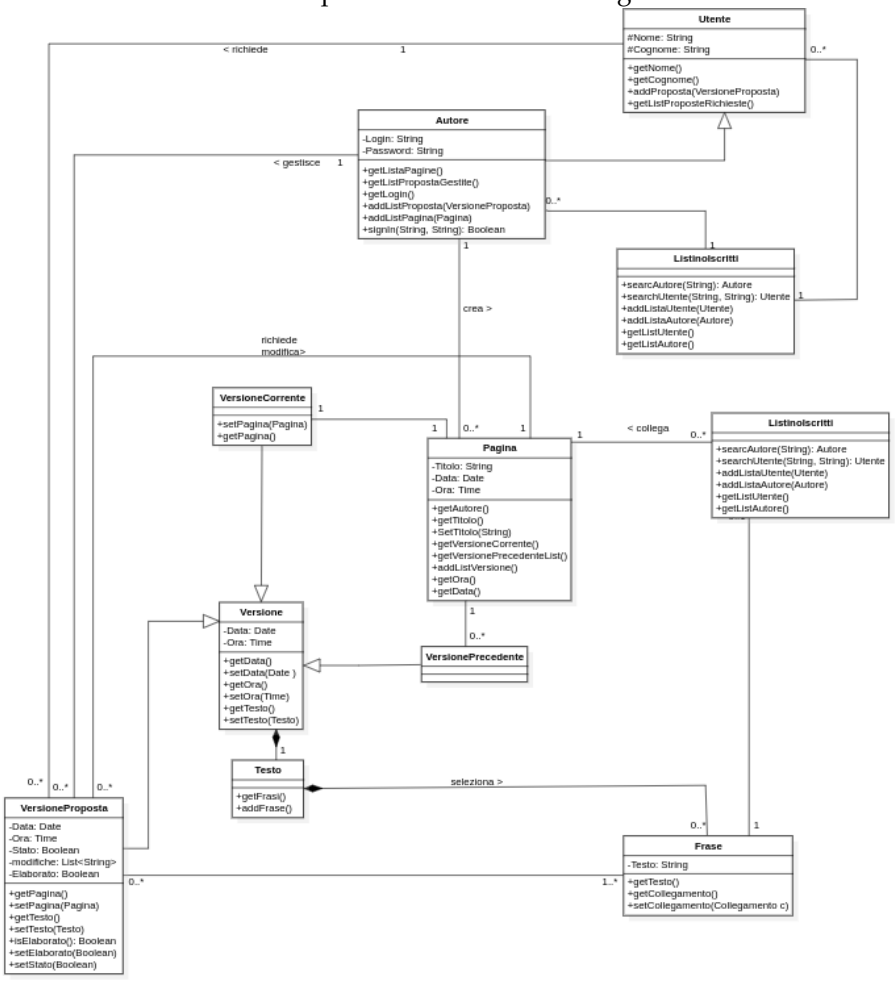
Ogni testo contiene frasi, quindi è importante il riferimento al testo stesso. Inoltre, una frase può fungere da collegamento verso un'altra pagina.

3.6 Collegamento

Questa classe contiene un riferimento alla pagina di destinazione e un riferimento alla frase che la collega.

3.7 Classe Aggiuntiva

ListinoIscritti: Questa classe è stata creata per contenere tutti gli iscritti al sistema, consentendo di effettuare le azioni necessarie a partire da autori o utenti generici.



4 Interfaccia Grafica

Durante lo sviluppo dell'interfaccia grafica, è stato utilizzato Swing come strumento principale.

4.1 Home

All'avvio del programma, l'utente è accolto da una schermata che presenta tre opzioni da scegliere: iscrizione, accesso come autore o accesso come utente.

4.2 Iscrizione

La schermata di iscrizione consente all'utente di registrarsi come utente generico inserendo solo nome e cognome, oppure come autore aggiungendo anche username e password.

4.3 Accesso Autore

L'accesso come autore è possibile attraverso questa schermata.

4.4 Accesso Utente

Gli utenti possono accedere al sistema tramite questa schermata.

4.5 Schermata Autore

In questa schermata, viene presentata una lista dei titoli delle pagine create dall'autore che ha effettuato l'accesso. L'autore può creare una nuova pagina, visualizzare le proposte da lui fatte o accedere alla schermata principale del sistema per visualizzare tutte le pagine e fare ulteriori proposte.

4.6 Creazione Pagina

La schermata di creazione pagina è accessibile dalla Schermata Autore e consente di inserire un titolo (controllando se è già in uso per un'altra pagina) e il testo della pagina. Ogni ritorno a capo indica una nuova frase. Dopo il salvataggio, l'autore viene reindirizzato alla schermata ViewPagina.

4.7 View Pagina

Questa schermata mostra il testo della pagina in una lista di frasi. L'autore può visualizzare le versioni precedenti e le proposte in ordine cronologico. Selezionando una frase, è possibile inserire un collegamento.

4.8 Create Link

La schermata Create Link si apre dalla ViewPagina e permette di inserire un collegamento nella frase selezionata, scegliendo la pagina di destinazione da un menu a tendina.

4.9 View Versioni Precedenti

Questa schermata è accessibile dalla ViewPagina e mostra una lista delle versioni precedenti della pagina, indicando data e ora di creazione per ciascuna versione.

4.10 Only View Page

La schermata Only View Page consente di visualizzare il contenuto di una versione senza poter apportare modifiche, selezionandola dalla lista delle versioni precedenti o dalle proposte.

4.11 View Proposte Autore

Questa schermata è resa visibile quando l'autore, dalla Schermata Autore, seleziona il tasto per visualizzare le pagine su cui ha proposto modifiche. Presenta una lista di tutte le proposte fatte dall'autore. Selezionando una proposta, si apre un popup menu che consente di visualizzare le versioni precedenti di tale pagina, la versione corrente o il testo della versione proposta dall'autore.

4.12 Schermata Utente

Dopo l'accesso di un utente, oppure quando un autore desidera visualizzare la schermata principale del sistema, viene mostrato il frame Schermata Utente che presenta una lista di titoli di tutte le pagine presenti nel sistema. È possibile filtrare i titoli inserendo delle sottostringhe nella JTextField.

4.13 View Pagina Utente

Selezionando un titolo dalla Schermata Utente, viene mostrato il frame View Pagina Utente che contiene il testo della versione corrente della pagina attraverso una lista di frasi. È possibile seguire i collegamenti e proporre modifiche alla pagina.

4.14 Creazione Proposta

La schermata di Creazione Proposta visualizza una lista delle frasi della pagina selezionata. Selezionando una frase dalla lista, si apre un menu popup che consente di modificare o eliminare la frase. Dopo il salvataggio, viene creata automaticamente la proposta.

4.15 View Proposte

La schermata View Proposte è accessibile dall'autore dalla View Pagina e mostra la proposta meno recente, ancora non elaborata, di tale pagina. L'autore può accettare o rifiutare la proposta e, se ci sono altre proposte, viene mostrata la successiva.

Per il model è presente nel file modelloGUI.mdj nella repository di github. Dovuto a scarsa visibilità.

5 Pattern DAO

5.1 AutoreDAO

Interfaccia che permette la manipolazione di dati riguardo l'entità autore nel database

5.1.1 ImplementazionePostgresAutoreDao

Implementa l'interfaccia AutoreDAO. Al momento della creazione dell'oggetto, apre la connessione al database. Ci sono metodi che attraverso una query:

1. Legge dal database gli autori esistenti
2. Inserisce un autore dal sistema, nel database

5.2 UtenteDao

Interfaccia che permette la manipolazione di dati riguardo l'entità utente nel database

5.2.1 ImplementazionePostgresUtenteDao

Implementa l'interfaccia AutoreDAO. Al momento della creazione dell'oggetto, apre la connessione al database. Ci sono metodi che attraverso una query:

1. Legge dal database gli utenti esistenti
2. Inserisce un utente nel database

5.3 CollegamentoDAO

Interfaccia riguardo l'entità collegamento

5.3.1 ImplementazionePostgresCollegamentoDAO

Implementa l'interfaccia CollegamentoDAO. Ci sono metodi che attraverso una query:

1. Legge dal database tutti i collegamenti e li inserisce correttamente nel sistema
2. Inserisce un collegamento appena inserito nel sistema, nel database
3. Attraverso un idTesto prende tutti i collegamenti presenti in quella pagina, dal database.

5.4 FraseDAO

Interfaccia riguardo l'entità frase

5.4.1 ImplementazionePostgresFrasiDAO

Implementa l'interfaccia FraseDAO. Ci sono metodi che attraverso una query:

1. Legge dal database tutti le frasi di pagine
2. Inserisce una frase nel database dato un titolo e la posizione nel testo
3. Inserisce una frase in una corretta versione proposta

5.5 ModificaDAO

Interfaccia riguardo l'entità modifica che si riferisce alla modifica proposta di una determianta frase selezioanta.

5.5.1 ImplementazionePostgresModificaDAO

Implementa l'interfaccia ModificaDAO. Ci sono metodi che attraverso una query:

1. Selezioanando una proposta e un idTesto della pagina da modificare, inserisce una modifica riguardo tale proposta nel database
2. data una proposta, seleziona dal database le frasi selezioante

5.6 PaginaDAO

Interfaccia riguardo l'entità pagina

5.6.1 ImplementazionePostgresPaginaDAO

Implementa l'interfaccia PaginaDAO. Ci sono metodi che attraverso una query:

1. Legge dal database tutti le pagina
2. Inserisce una pagina appena creata nel sistema, nel database

5.7 SelezionaDAO

Interfaccia riguardo l'entità seleziona che si riferisce alle frasi selezionate in una proposta.

5.7.1 ImplementazionePostgresSelezionaDAO

Implementa l'interfaccia SelezioaDAO. Ci sono metodi che attraverso una query:

1. Selezioanando una proposta e un idTesto della pagina da modificare, inserisce una frase selezionata nel database
2. data una proposta, carica nel sistema le frasi selezionate, attraverso la loro posiziona nel testo

5.8 TestoDAO

Interfaccia riguardo l'entità testo

5.8.1 ImplementazionePostgresTestoDAO

Implementa l'interfaccia TestoDAO. Ci sono metodi che attraverso una query:

1. ritorna il massimo valore che ha un idTesto nel database
2. inserisce nel database un testo

5.9 VersioneCorrente

Interfaccia riguardo l'entità versioneCorrente.

5.9.1 ImplementazionePostgresVersioneCorrenteDAO

Implementa l'interfaccia VersioneCorrenteDAO.

1. Aggiorna una versione corrente con un nuovo testo
2. dato un idTesto, restituisce il titolo della pagina
3. dato un titolo restituisce un idTesto

5.10 VersionePrecedente

Interfaccia riguardo l'entità versionePrecedente.

5.10.1 ImplementazionePostgresVersionePrecedenteDAO

Implementa l'interfaccia VersionePrecedenteDAO.

1. Legge dal database le versioni precedenti
2. Legge dal database per una versione precedente, le frasi che la compongono
3. Aggiunge una versione precedente

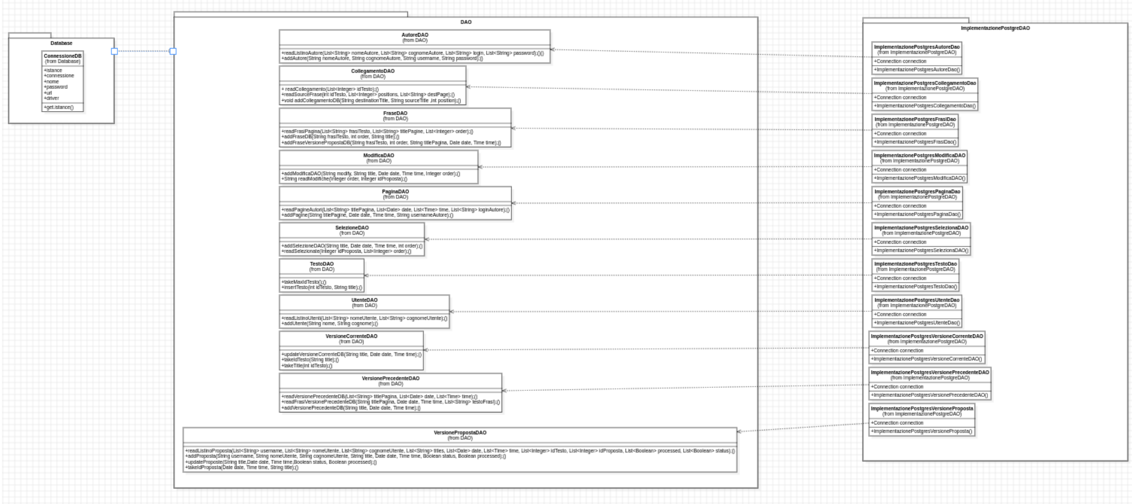
5.11 VersioneProposta

Interfaccia riguardo l'entità versioneProposta.

5.11.1 ImplementazionePostgresVersionePropostaDAO

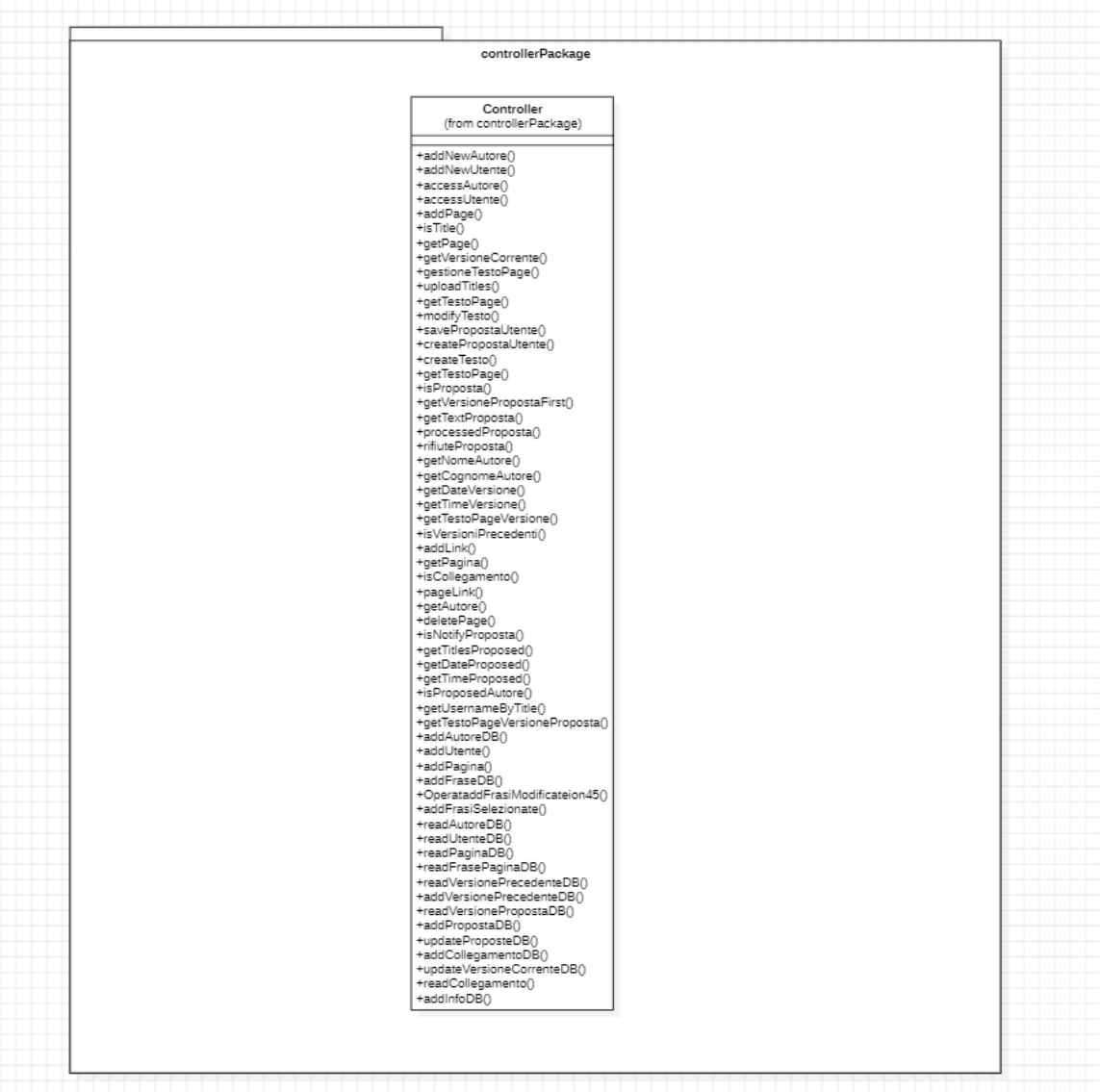
Implementa l'interfaccia VersionePropostaDAO.

1. Legge dal database le versioni proposte
2. Aggiorna lo stato di una proposta nel database
3. Aggiunge una versione proposta
4. Dato un titolo, una data e un orario ritorna l'idProposta

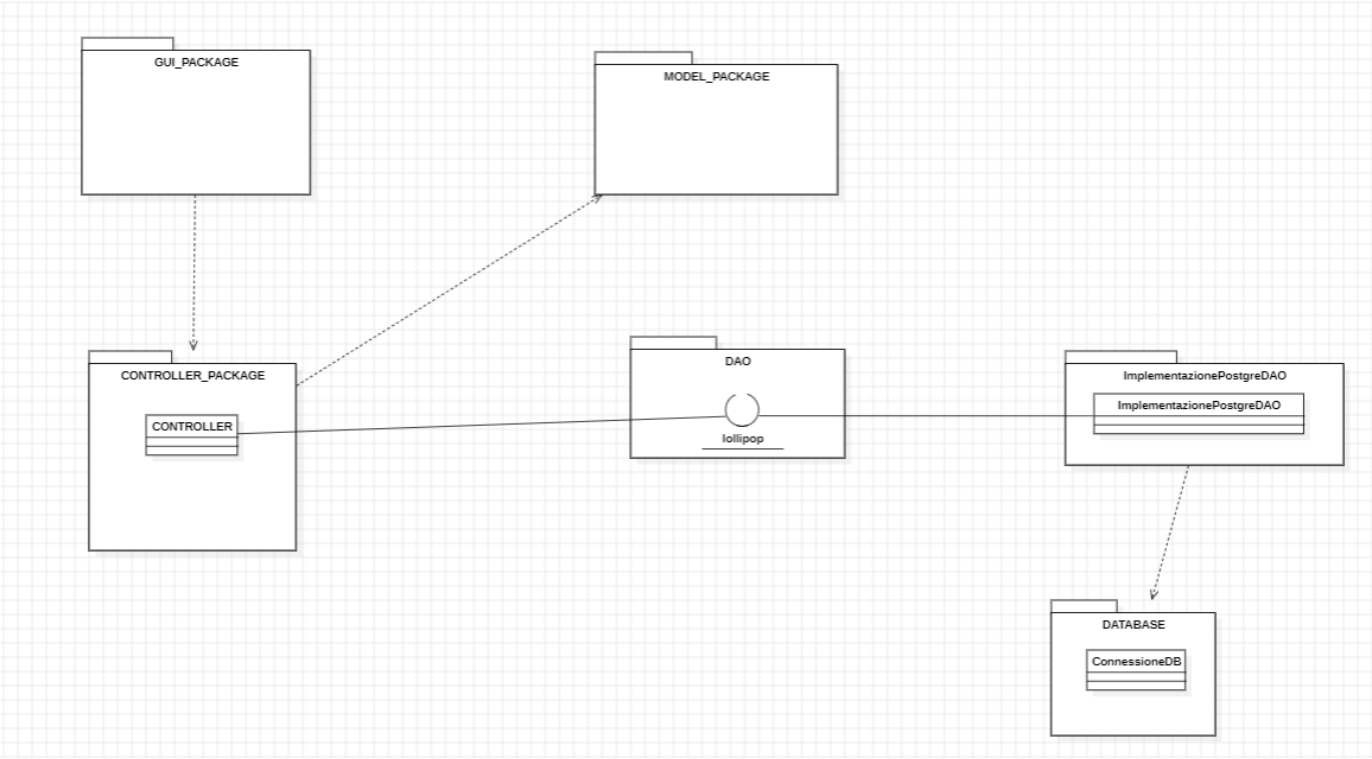

Database
ConnessioneDB
(from Database)
+istance
+connessione
+nome
+password
+url
+driver
+get.istance()
DAO
AutoreDAO
(from DAO)
+readListinoAutore(List<String> nomeAutore, List<String> cognomeAutore, List<String> login, List<String> password);()()
+addAutore(String nomeAutore, String cognomeAutore, String username, String password);()
CollegamentoDAO
(from DAO)
+ readCollegamento(List<Integer> idTesto);()
+readSourceFrase(int idTesto, List<Integer> positions, List<String> destPage);()
+void addCollegamentoDB(String destinationTitle, String sourceTitle ,int position);()
FraseDAO
(from DAO)
+readFrasiPagina(List<String> frasiTesto, List<String> titlePagine, List<Integer> order);()
+addFraseDB(String frasiTesto, int order, String title);()
+addFraseVersionePropostaDB(String frasiTesto, int order, String titlePagina, Date date, Time time);()
ModificaDAO
(from DAO)
+addModificaDAO(String modify, String title, Date date, Time time, Integer order);()
+String readModifiche(Integer order, Integer idProposta);()
PaginaDAO
(from DAO)
+readPagineAutori(List<String> titlePagina, List<Date> date, List<Time> time, List<String> loginAutore);()
+addPagine(String titlePagine, Date date, Time time, String usernameAutore);()
SelezioneDAO
(from DAO)
+addSelezioneDAO(String title, Date date, Time time, int order);()
+readSelezionate(Integer idProposta, List<Integer> order);()
TestoDAO
(from DAO)
+takeMaxIdTesto();()
+insertTesto(int idTesto, String title);()
UtenteDAO
(from DAO)
+readListinoUtenti(List<String> nomeUtente, List<String> cognomeUtente);()
+addUtente(String nome, String cognome);()
VersioneCorrenteDAO
(from DAO)
+updateVersioneCorrenteDB(String title, Date date, Time time);()
+takeIdTesto(String title);()
+takeTitle(int idTesto);()
VersionePrecedenteDAO
(from DAO)
+readVersionePrecedenteDB(List<String> titlePagina, List<Date> date, List<Time> time);()
+readFrasiVersionePrecedenteDB(String titlePagina, Date date, Time time, List<String> testoFrasi);()
+addVersionePrecedenteDB(String title, Date date, Time time);()
VersionePropostaDAO
(from DAO)
+readListinoProposta(List<String> username, List<String> nomeUtente, List<String> cognomeUtente, List<String> titles, List<Date> date, List<Time> time, List<Integer> idTesto, List<Integer> idProposta, List<Boolean> processed, List<Boolean> status);()
+addProposta(String username, String nomeUtente, String cognomeUtente, String title, Date date, Time time, Boolean status, Boolean processed);()
+updateProposte(String title,Date date, Time time,Boolean status, Boolean processed);()
+takeIdProposta(Date date, Time time, String title);()
ImplementazionePostgreDAO
ImplementazionePostgresAutoreDao
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresAutoreDao()
ImplementazionePostgresCollegamentoDao
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresCollegamentoDao()
ImplementazionePostgresFrasiDao
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresFrasiDao()
ImplementazionePostgresModificaDAO
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresModificaDAO()
ImplementazionePostgresPaginaDao
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresPaginaDao()
ImplementazionePostgresSelezionaDAO
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresSelezionaDAO()
ImplementazionePostgresTestoDao
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresTestoDao()
ImplementazionePostgresUtenteDao
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresUtenteDao()
ImplementazionePostgresVersioneCorrenteDAO
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresVersioneCorrenteDAO()
ImplementazionePostgresVersionePrecedenteDAO
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresVersionePrecedenteDAO()
ImplementazionePostgresVersioneProposta
(from ImplementazionePostgreDAO)
+Connection connection
+ImplementazionePostgresVersioneProposta()

6 Controller

Come assegnato, abbiamo usato un unico controller per gestire tutte le operazione e la gestione dei dati del model. Sono presenti quindi tutti i metodi che richiedono informazioni dal model e dal database ed eseguono operazioni richieste dalla GUI così rispettando il modello scelto BCE+PatternDAO.



7 Modello soluzione del problema

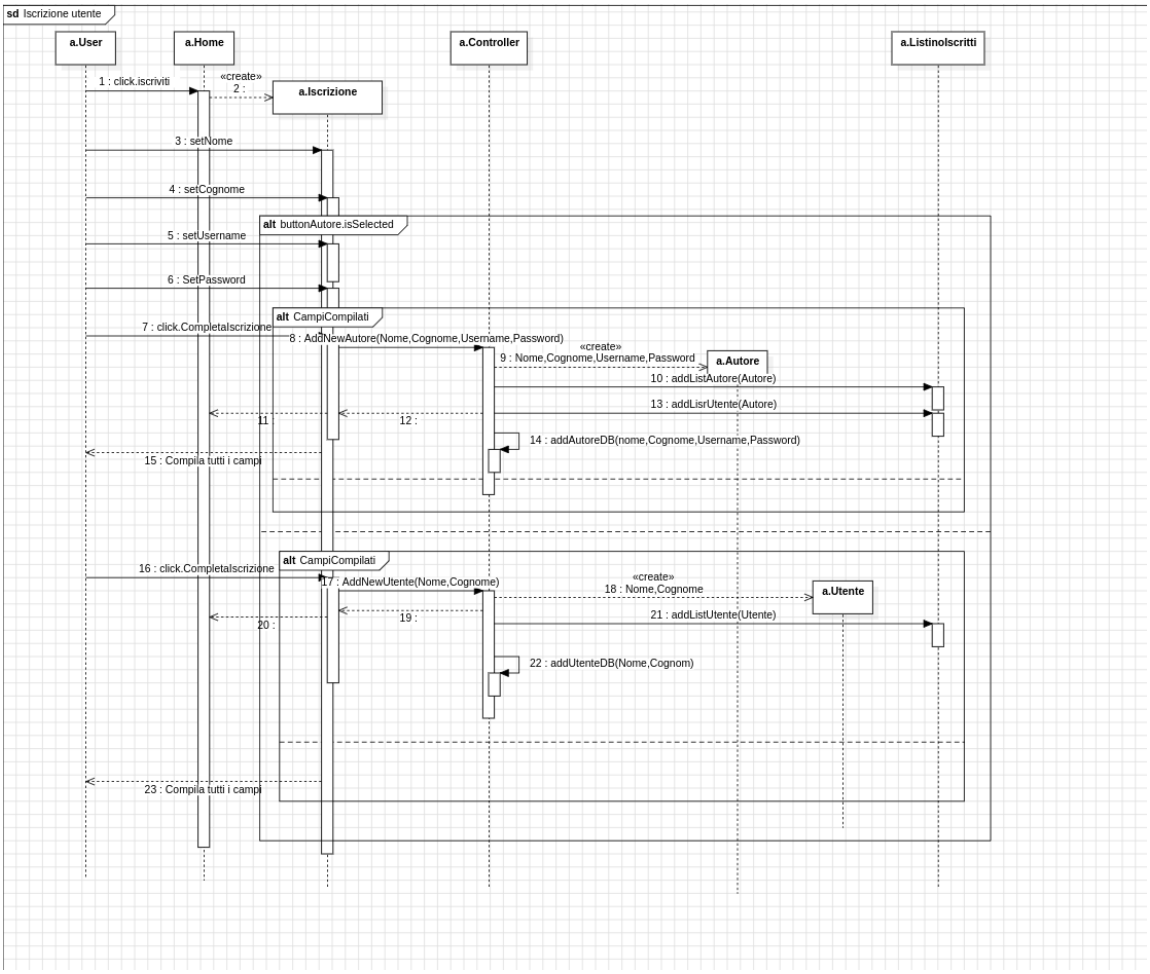


8 SEQUENCE DIAGRAM

Abbiamo scelto di rappresentare attraverso sequence diagram, due funzionalità:

1. Iscrizione di un utente
2. Creazione di una pagina

8.1 Iscrizione di un utente

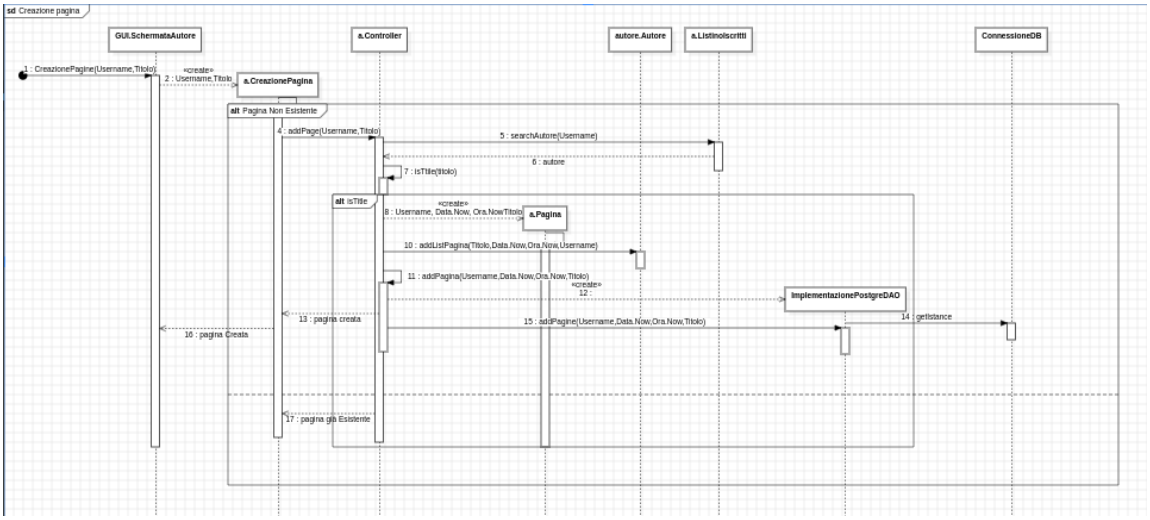


Quando un utente avvia il sistema, viene presentata la schermata principale, la Home. Per iscriversi, l'utente premerà il pulsante "Iscriviti", il che aprirà la schermata di Iscrizione. Qui, l'utente inserirà il suo nome e cognome. Se entrambi i campi sono compilati, il metodo **addNewUtente** presente nel controller associato alla schermata di Iscrizione consentirà la creazione di un nuovo oggetto Utente, a condizione che non sia già iscritto. Questo nuovo utente sarà quindi memorizzato nel database.

Se l'utente desidera diventare un autore per creare pagine, dovrà selezionare la casella di controllo dedicata. Ciò consentirà di inserire un nome utente e una password. Se tutti i campi pertinenti sono compilati, il sistema chiamerà il metodo **addNewAutore** presente nel controller associato alla schermata di Iscrizione. Questo metodo creerà un nuovo oggetto Autore, che verrà quindi inserito nel database.

Sia che l'utente si registri come utente generico o come autore, al momento dell'iscrizione sarà aggiunto alla lista degli iscritti utilizzando un'apposita lista gestita dall'applicazione.

8.2 Creazione pagina



Nella sequenza di azioni per la creazione di una pagina, si parte dalla schermata dell'autore per poi passare alla schermata di creazione della pagina. Qui, inserendo un titolo, viene richiamato il metodo **addPage** presente nel controller.

Inizialmente, il sistema utilizza l'username per ottenere l'oggetto autore tramite il metodo **searchAutore** presente in ListinoIscritti. Se il titolo inserito non è già utilizzato per un'altra pagina, viene creato un nuovo oggetto pagina. Questa pagina viene quindi inserita nella lista delle pagine dell'autore attraverso il metodo **addListAutore**.

Successivamente, questa pagina viene caricata nel database mediante l'uso del metodo **addPagina** presente nel controller. Tale metodo istanzia un oggetto **ImplementazionePostgresPaginaDAO** che apre una connessione. A questo punto, il controller richiama il metodo **addPagine** in ImplementazionePostgresPaginaDAO, il quale aggiunge la nuova pagina al database.