Università degli Studi
di Napoli Federico II

Documentazione Sistema Di Gestione Del Ciclo Di Vita Di Una Pagina Wiki

Giuseppe Pollio, Mario Lombardo

Anno accademico 2023-2024

Indice

1 Dominio del problema

1.1 Introduzione

In questa soluzione, esaminiamo attentamente il problema per identificarne il dominio, con l'intento di sviluppare un diagramma che orienterà le nostre decisioni di progettazione a livello di codice.

1.2 Analisi dei requisiti

In questa sezione si analizza la specifica con lo scopo di definire le funzionalità che la base di dati deve soddisfare. Individueremo le entità e le associazioni del mini-word, e produrremo una prima versione del modello concettuale che sarà poi rielaborato nelle fasi successive della progettazione.

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

L'introduzione individua il mini-world da rappresentare e una prima entità: **Page** avente come attributi **title** e **creation**. Inoltre nel testo della **Page** possiamo individuare un'altra entità associata alla pagina (**Page**): **PageText** contenente come attributi la riga di testo (**text**), un collegamento ad un'altra pagina **link** tramite attributo parziale e una relazione all'entità **Page**.

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. Una frase può contenere anche un collegamento. Ogni collegamento è caratterizzato da una frase da cui scaturisce il collegamento e da un'altra pagina destinazione del collegamento.

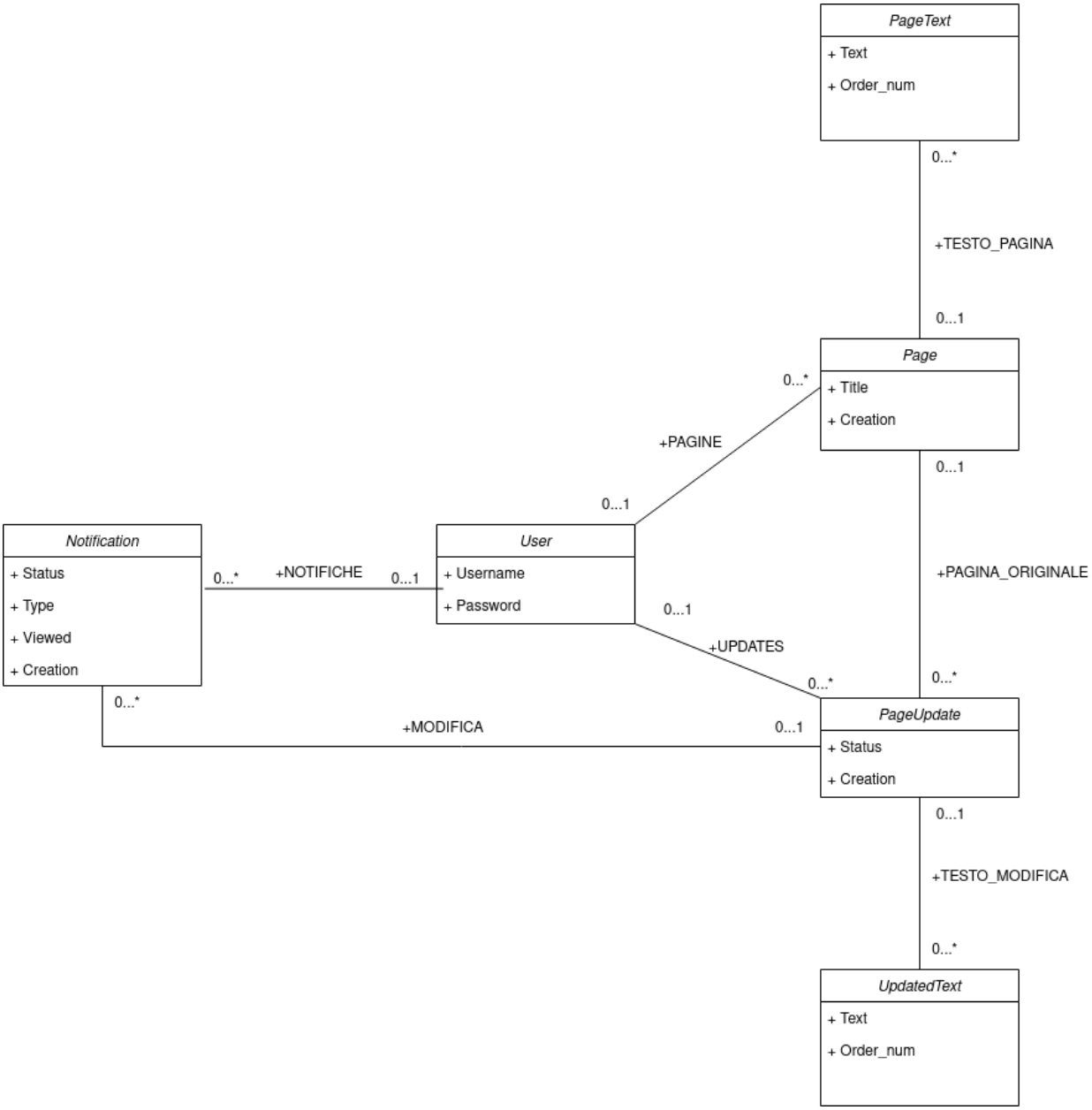
Dato che le pagine delle wiki saranno publiche oltre che modificabili, sarà necessario tenere traccia degli utenti tramite l'entità **User**, alla quale sarà possibile accederci tramite una **password** e un **username**. Di conseguenza all'entità **Page** viene inserito individuato l'autore della pagina tramite il campo **author**. Le modifiche del testo di una pagina (**Page**) saranno salvate tramite le l'entità **PageUpdate** avente attributi **status** che indica lo stato della modifica e un collegamento ad **User** che rappresenta l'autore della modifica (**author**). Il testo modificato viene individuato tramite l'entità **UpdatedText** contenente il "nuovo testo" individutato dall'attributo **text**, un eventuale **link** (collegamento) tramite attributo parziale e un collegamento all'entità **PageUpdate**, quando una modifica viene inoltrata all'autore esso invece deve essere avvisato tramite una notifica individuata dall'entità **Notification** avente come attributi **status** (stato di lettura), **type**, un collegamento a **User** per individuare il proprietario della notifica e un collegamento a **PageUpdate**.

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).

Viene in aggiunta inserito il nuovo attributo **order_num** alle entità **PageText** e **UpdatedText** che rappresenta l'ordinamento all'interno del testo della **Page**. Inoltre si farà la distinzione tra un Utente (User) e Amministratore (Admin) tramite specializzazione parziale.

1.3 Diagramma del Dominio del Problema

In seguito alle considerazioni espresse nella sezione precedente, si è prodotto il seguente class diagram del dominio del problema:



1.4 Dizionari

1.4.1 Dizionario delle Entità

Entità	Descrizione	Attributi
User	Account Utente della wiki.	**Username**(Stringa): nome identificativo dell'account utente. **Password**(Stringa): password necessaria per accedere all'account utente.
Page	Pagina presente sulla wiki.	**Title**(Stringa): Titolo della pagina. **Creation**(Data): Data e ora di creazione della pagina.
PageText	Frase di una Pagina (**Page**).	**Text**(Stringa): Contenuto della frase. **Link**(Stringa): Collegamento della frase (interno o esterno alla wiki). **Order_num**(Intero): Ordinamento della frase all'interno del testo complessivo.
PageUpdate	Modifica proposta ad pagina da parte di un altro utente.	**Status**(Intero): Stato della richiesta di modifica.
UpdatedText	Nuovo testo proposto durante la modifica (**PageUpdate**).	**Text**(Stringa): Contenuto della frase. **Link**(Stringa): Collegamento della frase (interno o esterno alla wiki). **Order_num**(Intero): Ordinamento della frase all'interno del testo complessivo.
Notification	Notifiche di un **Utente (User)**.	**Status**(Booleano): Stato di lettura di una notifica. **Type**(Intero): Tipologia di notifica.

1.4.2 Dizionario delle Associazioni

Associazione	Tipologia	Descrizione
Autore Pagina	uno-a-molti	Associa ad ogni pagina (Page) un utente (User) che ne rappresenta l'autore
Autore Modifica	uno-a-molti	Associa ad ogni modifica (PageUpdate) un utente (User) che ne rappresenta l'autore
Testo Pagina	uno-a-molti	Associa ad ogni pagina (Page) tutto il testo (PageText) appartenente a quella determinata pagina.
Testo Modifica	uno-a-molti	Associa ad ogni modifica (PageUpdate) tutto il testo (UpdatedText) appartenente a quella determinata modifica.
Proprietario Notifica	uno-a-molti	Associa ad ogni notifica (Notification) un utente (User) che ne rappresenta l'autore.
PageUpdate Notifica	uno-a-molti	Associa ad ogni notifica (Notification) una modifica (PageUpdate) che ne aiuta a rappresentare il contenuto.
Page PageUpdate	uno-a-molti	Associa ad ogni Modifica proposta (PageUpdate), La pagina (Page) per cui è stata proposta.

2 Impementazione e Dominio della Soluzione

In questa sezione, illustreremo l'architettura dell'applicazione e i vari pattern utilizzati, fornendo occasionali annotazioni implementative. Abbiamo scelto di escludere la trattazione del codice effettivo, il quale è disponibile nella documentazione JavaDoc del codice sorgente e, naturalmente, nel sorgente stesso.

2.1 Architettura BCED

L'applicazione è stata progettata seguendo i principi dell'architettura *Boundary-Control-Entity con estensione Database*, nota come BCED. Il pattern BCED fornisce una struttura ben organizzata per separare le diverse responsabilità all'interno di un software, basandosi su quattro componenti principali:

- **Boundary**: Questa componente gestisce ogni interazione con agenti esterni. Nel contesto di un'applicazione con interfaccia grafica, la boundary corrisponde all'interfaccia stessa, e l'agente esterno è l'utente. Essa consente all'utente di comunicare con l'applicazione e comunica a sua volta con il resto del programma attraverso il controller.

  Nel nostro codice sorgente, la boundary è implementata nel package GUI.

- **Control**: Questa componente è al centro dell'applicazione, mettendo in comunicazione tutte le sue parti ed eseguendo la logica di business. Né la Boundary né l'Entity sono autorizzate a comunicare se non tramite il Control. Nel nostro codice sorgente.

  Il control è implementato nel package **Controllers**.

- **Entity**: Per Entity si intende l'insieme delle informazioni da memorizzare durante l'esecuzione che sono di interesse nel dominio. Nel contesto di un'applicazione, essa è rappresentata dal modello di dati interno. Nel nostro codice sorgente.

  L'Entity è implementata nel package **Models**.

- **Database**:

  Costituisce la sorgente esterna di dati a lungo termine nell'ambito del nostro sistema. In questo contesto specifico, si tratta di un database Postgres. La connessione a tale database è implementata nel nostro codice attraverso due package: **DAO** e **Database**. Per un'analisi più dettagliata del pattern DAO, si rimanda alla sezione apposita.

2.2 Controller

3 Overview dell'Applicazione

Dopo aver esaminato l'implementazione dell'applicazione, procediamo ora alla discussione dell'applicazione stessa. Questa sezione offre una panoramica, finestra per finestra, dell'applicazione e delle sue funzionalità.

3.0.1 LoginPage

L'applicazione ha come sua view iniziale la **LoginPage**. Qui l'utente può decidere se continuare col proprio account, inserendo i propri dati, registrarsi aprendo la view **RegisterPage** e infine accedere alla wiki come ospite.

3.0.2 RegisterPage

In questa pagina è possibile creare un nuovo utente oppure tornare indietro al login (**LoginPage**)
Il primo utente ad iscriversi riceve un account col ruolo di *Amministratore*.

3.0.3 MainMenu

Questa è la pagina principale della wiki, da qui è possibile:

- Leggere le proprie notifiche (se l'utente è loggato altrimenti accedere/registrarsi).
- Cercare le pagine per testo o per autore.
- Accedere a tutte le pagine presenti sulle wiki.
- Creare una nuova pagina.

3.0.4 PageView

Da qui è possibile visualizzare e compiere alcune azioni per una pagina presente sulla wiki.

È possibile eseguire azioni del tipo:

- Avviare una richiesta di modifica la pagina (oppure modificarla direttamente se si è amministratori o proprietari della pagina)
- Eliminare la pagina se si è amministratori o proprietari della pagina

3.0.5 PageCreate

Qui l'utente può creare nuove pagine attraverso l'editor testuale.

3.0.6 PageEdit

Qui l'utente può modificare una pagina tramite l'editor testuale.

3.0.7 PageHistory

Pagina in cui è possibile visualizzare la storia delle modifiche di una pagina

3.0.8 UserNotifications

Da questa pagina è possibile visualizzare le proprie notifiche e accedere alla pagina del confronto modifiche

3.0.9 ReviewUpdate

L'utente da questa pagina può confrontare le modifiche proposte ad una pagina e decidere se rifiutarle o meno