Relazione Basi di Dati

Giada De Paolis mt. 615012

January 9, 2025

1 Relazione

Il progetto realizzato è una piattaforma per la gestione di contenuti (blog, post, commenti, like, ecc.). Per la progettazione dell'interfaccia utente è stato usato Taliwind CSS, mentre il backend è stato implementato in PHP. La decisione di utilizzare Tailwind CSS è data dalla flessibilità di creare design personalizzati, è un framework che consente di ridurre i tempi di sviluppo, eliminando la necessità di scrivere fogli di stile CSS. I file PHP includono funzionalità per la creazione, modifica ed eliminazione di contenuti (blog, post, commenti, profilo utente). Nel dettaglio

- il file db_connect.php gestisce la connessione al database
- create_blog.php, edit_blog.php e delete_blog.php sono file dedicati alla gestione dei blog: creazione, modifica e eliminazione
- stesso discorso per i file edit_profile.php, delete_profile.php per quanto riguarda la modifica e l'eliminazione dell'account
- anche per i post è stata usata la stessa logica nei file post.php (dove c'è una sezione dedicata alla creazione di nuovi post) e delete_post.php per l'eliminazione dei post, non c'è una sezione dedicata per la modifica dei post

Tutti gli errori vengono gestiti dalla pagina errors.php.
Un utente accede per prima cosa alla pagina registration.php: inserisce username (unico, non ci posso essere account con lo stesso username), name, surname, email (unico come surname), date of birth, password e password confirm, quest'ultimo non è presente nel dabase, per non occupare spazio in memoria inutilmente.
Dopo aver completato la registrazione, veniamo mandati nella schermata di Log In login.php, dove inserendo username e password possiamo accedere al blog, venendo reindirizzati alla pagina principale index.php, chiamato Feed. La pagina index è visibile anche agli user non loggati. Nella pagina index sono presenti tutti i blog creati in ordine dal piu recente al meno recente.
Nella pagina head.php vengono distinti utenti loggati e non, i primi vedranno

tramite pagina nav_auth.php la navbar con i tasti di logout e 'go premium' per coloro che non sono ancora premium, mentre i non loggati vedranno solo il tasto di login tramite pagina nat_unauth.php, tramite tasto login saremmo indirizzati alla schermata di log in, in fondo al box c'è la possibilità di andare nella schermata di registrazione se non si ha ancora un account.
Il tasto 'go premium' sparirà dopo che l'user sarà passato a premium tramite la pagina premium.php, bisogna compilare tutti i dati richiesti e una volta che i campi sono non vuoti e non ci sono errori l'utente avrà lo status is_premium==true; i dati del pagamento sono fasulli, non sono collegati al database e non c'è una tabella dedicata agli utenti premium. Una volta passato a premium l'user potrà creare tutti i blog che vuole, mentre se si è un user regolare si potranno creare soltanto un massimo di 3 blogs.
Un utente loggato potrà visualizzare i propri blog e i blog dove è coautore nella pagina myblogs.php, da lì tramite tasto crea blog, si potranno creare nuovi blog. Nel momento della creazione di un nuovo blog andranno specificati titolo, descrizione, categoria (tramite dropdown, dove c'è una sezione dedicata alla creazione di nuove categorie), in base alla categoria selezionata, nel dropdown della sottocategorie, saranno visibili tutte le sottocategorie appartenenti. Allo stesso modo si potranno creare nuove sottocategorie. Opzionalmente si può aggiungere una foto.
Una volta creato il nuovo blog, verremo indirizzati alla pagina myblogs.php. Se siamo i proprietari del blog, saranno visibili i tasti delete e edit, tramite tasto edit, oltre a poter modificare titolo e descrizione, è possibile aggiungere uno o più coautori, tramite dropdown, dove sono visibili tutti gli utenti registrati e basterà cliccare sul nome dell'utente che si vuole aggiungere.
Tornando alla pagina myblogs, cliccando sul titolo si accederà al contenuto del blog, ossia tutti i post. Il proprietario del blog e i coautori potranno creare post. Dopo la sezione dedicata per la pubblicazione dei post, è visibile la lista di tutti i post creati per quel blog, dove si potrà aggiungere like o creare un commento (che può essere eliminato dal proprietario del commento). Il tasto remove post è visibile sempre al proprietario del blog e ai coautori solo se hanno creato il post. Quando si crea un nuovo post, andranno specificati titolo, descrizione e si potranno caricare più immagini.
Nella pagina profile.php, chiamata myprofile, saranno visibili i dettagli dell'utente, i quali possono essere modificati tramite tasto edit e si potrà eliminare il proprio profilo tramite tasto delete.
Infine, tramite tasto logout è possibile disconnettere l'account.

2 Schema Logico

- gli attributi sottolineati sono PRIMARY KEY
- gli attributi con * sono FOREIGN KEY

BLOG_(id, title, description, image, created_at, ∗id_user, ∗id_category, ∗id_subcategory)
BLOG_COAUTHOR(∗id_blog, ∗id_user)
CATEGORY(id, name)
COMMENT(id, text, ∗id_user, ∗id_post, created)
LIKE_(∗id_user, ∗id_post)
POST_(id, title, content, creted_at, ∗id_user, ∗id_blog)
POST_IMAGE(id, image_path, ∗post_id)
SUBCATEGORY(id, name, ∗id_category)
USER(id, username, name, surname, email, date_of_birth, password, is_premium)

Il presente schema logico presenta alcune ridondanze:

- nella tabella BLOG_ ho chiamato anche l'id delle subcategory nonostante possa estrarlo dalla tabella category, tramite foreign key id_category nella colonna della tabella BLOG_
- la tabella BLOG_COAUTHOR semplicemente è una relazione che prende l'id del blog e l'id dell'user
- stesso ragionamento con la tabella LIKE_, prendo l'id_user e l'id_post
- la tabella POST_IMAGE ha una relazione semplice con la tabella POST_, utile per poter caricare più di una immagine quando si crea un nuovo post
- le relazioni molti-a-molti sono gestite con tabelle ponte, come BLOG_COAUTHOR, per evitare duplicazioni e mantenere una struttura scalabile
- ogni SUBCATEGORY appartiene ad una sola CATEGORY, quindi è una relazione gerarchica
- tutti gli ID sono autoincrement, e tutte le date sono del tipo CURRENT_TIMESTAMP()

3 Diagramma ER

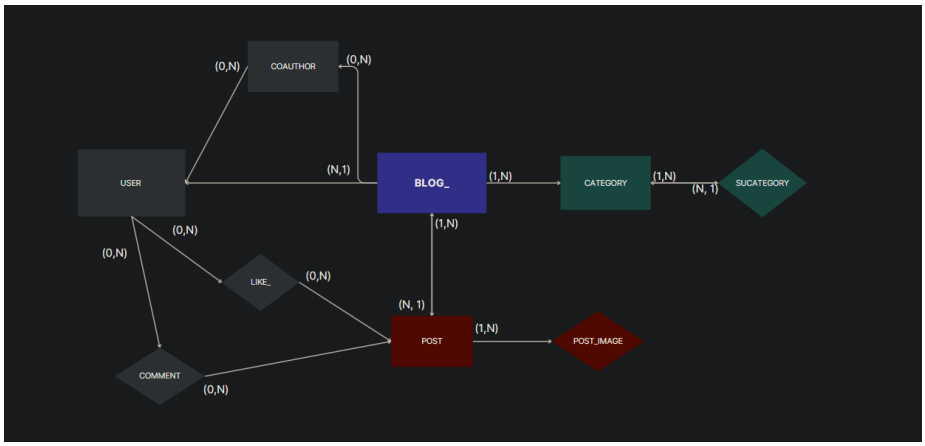

Figure 1: Diagramma ER