Università di Pisa

Relazione Progetto

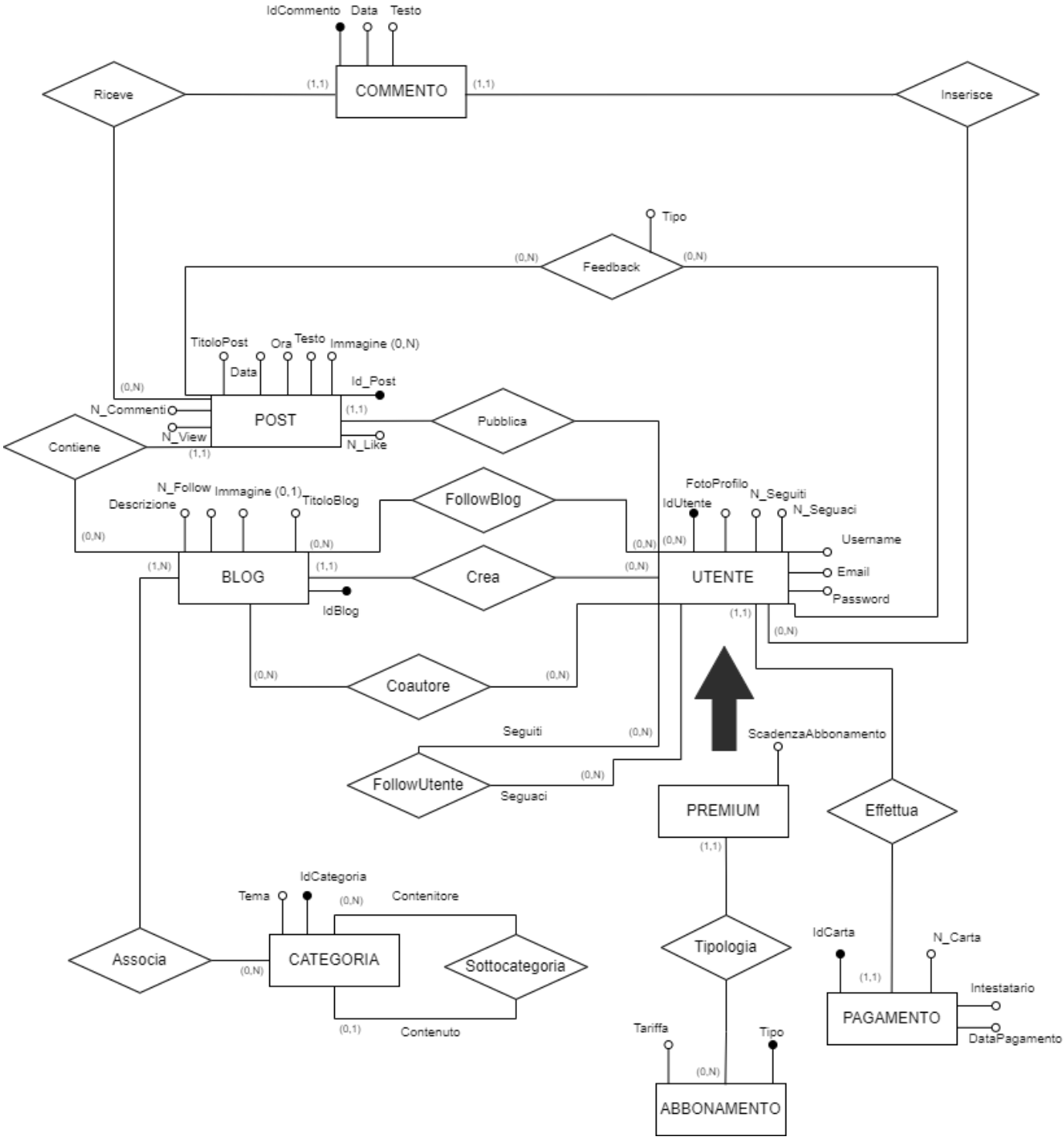
Basi di Dati e Laboratorio di Programmazione Web
2023/2024

Andrea Belliani 596864

Indice

1 Progettazione Concettuale

1.1 Diagramma E-R



1.1.1 Note

La tabella Feedback rappresenta l'interazione di un Utente con un Post. L'attributo Tipo, rappresentato da due valori, indica una visualizzazione (0) o un mi piace (1).

1.2 Dizionario delle Entità

Entità	Descrizione	Attributi	Identificatore
Utente	Individuo iscritto al servizio	IdUtente, Username, Email, Password, N_Seguaci, N_Seguiti, FotoProfilo	IdUtente
Premium	Utente abbonato al servizio	IdUtente, Tipo, ScadenzaAbbonamento	IdUtente
Abbonamento	Importo dovuto per abbonarsi al servizio	Tipo, Tariffa	Tipo
Pagamento	Metodo con il quale l'utente può abbonarsi al servizio	N_Carta, IdUtente, Intestatario, DataPagamento	N_Carta
Blog	Tipo di sito web nel quale l'utente può creare e gestire contenuti	IdBlog, IdUtente, TitoloBlog, Desrizione, Immagine, N_Follow	IdBlog
Post	Contenuto di un blog pubblicato da un utente	IdPost, IdUtente, IdBlog, TitoloPost, Testo, Data N_Like, N_Commenti, N_View	IdPost
Categoria	Argomento affrontato da un blog	IdCategoria, Tema	IdCategoria
Commento	Risposta che l'utente può lasciare sotto un post	IdCommento, Data, Testo	IdCommento

1.3 Dizionario delle Relazioni

Relazioni	Descrizione	Componenti	Attributi
Effettua	Utente effettua un pagamento	Utente, Pagamento	
Tipologia	Tipo di abbonamento per i premium	Premium, Abbonamento	
Crea	Utente crea un blog	Utente, Blog	
Associa	Categoria associata ad un blog	Blog, Categoria	
Sottocategoria	Categoria figlia di un'altra categoria	Categoria, Categoria	
Coautore	Utente scelto per gestire blog dell'autore	Utente, Blog	
FollowUtente	Utente segue altri utenti	Utente, Utente	
FollowBlog	Utente segue blog	Utente, Blog	
Contiene	Post appartiene ad un blog	Blog, Post	
Pubblica	Utente pubblica contenuti	Utente, Post	
Inserisce	Utente lascia un commento ad un post	Commento, Utente	
Riceve	Commenti ricevuti ad un post	Commento, Post	
Feedback	Utente lascia un mi piace o visualizza un contenuto	Utente, Post	Tipo

2 Progettazione Logica

2.1 Analisi delle ridondanze

Per l'analisi delle ridondanze è necessario calcolare il costo delle singole operazioni attraverso la formula generale:

$$C(O) = F(O) \cdot wT \cdot (a \cdot \text{NAw} \cdot \text{NAr})$$

- $F(O)$: frequenza dell'operazione
- wT : coefficiente correttivo che varia in base al tipo dell'operazione
- a : coefficiente moltiplicativo relativo al costo di una scrittura
- NAw : numero di accessi in scrittura
- NAr : numero di accessi in lettura

Dopo aver calcolato le operazioni si calcola il costo dello schema $C(S)$ e dello schema con ridondanza $C(S_{\text{rid}})$:

$$C(S) = \sum_{i=1}^{n} C(O_i)$$

e l'occupazione di memoria $M(S)$ e occupazione di memoria con ridondanza $M(S_{\text{rid}})$ rappresentata in byte indicando i volumi di entità e relazioni e grandezza di ciascun attributo di entità e relazione:

$$M(S) = \sum_{i=1}^{n} V(e_i) \cdot \text{size}(e_i) + \sum_{i=1}^{n} V(r_j) \cdot \text{size}(r_j)$$

2.1.1 N_Seguaci, N_Seguiti

Tavola delle Operazioni

- **Operazione 1**: utente segue altro utente
  - **Tipo**: Interattiva
  - **Frequenza**: 10 volte al giorno
- **Operazione 2**: visualizza seguiti di un utente
  - **Tipo**: Interattiva
  - **Frequenza**: 100 volte al giorno

Tavola dei Volumi

Concetto	Tipo	Volume
Utente	E	5
FollowUtente	R	20

Con Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Utente	Entità	2	R
FollowUtente	Relazione	1	W
Utente	Entità	2	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 3 + 2) = 80$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Utente	Entità	1	R

$$C(Op_2) = 100 \cdot 1 \cdot (2 \cdot 0 + 1) = 100$$

Senza Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
FollowUtente	Relazione	1	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 1 + 0) = 20$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
FollowUtente	Relazione	20	R

$$C(Op_2) = 100 \cdot 1 \cdot (2 \cdot 0 + 20) = 2000$$

Costo Dello Schema

$$C(S) = C(Op1) + C(Op2) = 20 + 2000 = 2020$$

$$C(S_{\text{rid}}) = C(Op_1) + C(Op_2) = 80 + 100 = 180$$

$$\frac{C(S)}{C(S_{\text{rid}})} = \frac{2140}{180} \approx 11.89$$

Occupazione di Memoria

$$M(S) = 412 * 5 + 8 * 20 = 2060 + 160 = 2220 \text{ bytes}$$

$$M(S_{\text{rid}}) = 420 \cdot 5 + 8 \cdot 20 = 2100 + 160 = 2260 \text{ bytes}$$

$$M(S) - M(S_{\text{rid}}) = 2220 - 2260 = -40 \text{ bytes}$$

In conclusione: Il costo con ridondanza è significativamente inferiore rispetto al costo senza ridondanza. La differenza di utilizzo della memoria è minima, con la ridondanza che utilizza solo 40 byte in più.

2.1.2 N_Follow

Tavola delle Operazioni

- **Operazione 1**: utente segue blog
  - **Tipo**: Interattiva
  - **Frequenza**: 10 volte al giorno
- **Operazione 2**: visualizzare i seguiti di un blog
  - **Tipo**: Interattiva
  - **Frequenza**: 100 volte al giorno

Tavola dei Volumi

Concetto	Tipo	Volume
FollowBlog	R	30
Blog	E	25

Con Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
FollowBlog	Relazione	1	W
Blog	Entità	1	R
Blog	Entità	1	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 2 + 1) = 50$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Blog	Entità	1	R

$$C(Op_2) = 100 \cdot 1 \cdot (2 \cdot 0 + 1) = 100$$

Senza Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
FollowBlog	Relazione	1	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 1 + 0) = 20$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
FollowBlog	Relazione	30	R

$$C(Op_2) = 100 \cdot 1 \cdot (2 \cdot 0 + 30) = 3000$$

Costo Dello Schema

$$C(S) = C(Op_1) + C(Op_2) = 20 + 3000 = 3020$$
$$C(S_{\text{rid}}) = C(Op_1) + C(Op_2) = 50 + 100 = 150$$
$$\frac{C(S)}{C(Srid)} = \frac{3140}{160} \approx 19.62$$

Occupazione di Memoria

$$M(S) = 551 \cdot 25 + 8 \cdot 30 = 13775 + 240 = 14015Byte$$
$$M(Srid) = 555 \cdot 25 + 8 \cdot 30 = 13875 + 240 = 14115 \text{ Byte}$$
$$M(S) - M(S_{\text{rid}}) = 14015 - 14115 = -100 \text{ byte}$$

In conclusione: Il costo con ridondanza è significativamente inferiore rispetto al costo senza ridondanza. La differenza di utilizzo della memoria è minima, con la ridondanza che utilizza solo 100 byte in più.

2.1.3 N_Like, N_View

Si applica sugli stessi dati

Tavola delle Operazioni

- **Operazione 1**: utente mette mi piace ad un post
  - **Tipo**: Interattiva
  - **Frequenza**: 10 volte al giorno
- **Operazione 2**: visualizzare i like/view di un post
  - **Tipo**: Interattiva
  - **Frequenza**: 100 volte al giorno

Tavola dei Volumi

Concetto	Tipo	Volume
Feedback	R	100
Post	E	50

Con Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Feedback	Relazione	1	W
Post	Entità	1	R
Post	Entità	1	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 2 + 1) = 50$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Post	Entità	1	R

$$C(Op_2) = 10 \cdot 1 \cdot (2 \cdot 0 + 1) = 10$$

Senza Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Feedback	Relazione	1	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 1 + 0) = 20$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Feedback	Relazione	100	R

$$C(Op_2) = 10 \cdot 1 \cdot (2 \cdot 0 + 100) = 1000$$

Costo Dello Schema

$$C(S) = C(Op_1) + C(Op_2) = 20 + 1000 = 1020$$

$$C(S_{\text{rid}}) = C(Op_1) + C(Op_2) = 50 + 10 = 60$$

$$\frac{C(S)}{C(S\text{rid})} = \frac{1020}{60} = 17$$

Occupazione di Memoria

$$M(S) = 9 \cdot 100 + 316 \cdot 50 = 900 + 15800 = 16700 \text{ byte}$$
$$M(S_{\text{rid}}) = 9 \cdot 100 + 320 \cdot 50 = 900 + 16000 = 16900 \text{ byte}$$
$$M(S) - M(S_{\text{rid}}) = 16700 - 16900 = -200 \text{ byte}$$

In conclusione: Il costo con ridondanza è significativamente inferiore rispetto al costo senza ridondanza. La differenza di utilizzo della memoria è minima, con la ridondanza che utilizza solo 200 byte in più.

2.1.4 N_Commenti

Tavola delle Operazioni

- **Operazione 1**: utente commenta un post
  - **Tipo**: Interattiva
  - **Frequenza**: 10 volte al giorno
- **Operazione 2**: visualizzare i commenti di un post
  - **Tipo**: Interattiva
  - **Frequenza**: 100 volte al giorno

Tavola dei Volumi

Concetto	Tipo	Volume
Commento	R	100
Post	E	50

Con Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Commento	Relazione	1	W
Post	Entità	1	R
Post	Entità	1	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 2 + 1) = 50$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Post	Entità	1	R

$$C(Op_2) = 10 \cdot 1 \cdot (2 \cdot 2 + 1) = 50$$

Senza Ridondanza

- **Operazione 1**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Commento	Relazione	100	W

$$C(Op_1) = 10 \cdot 1 \cdot (2 \cdot 1 + 0) = 20$$

- **Operazione 2**
  PARAMETRI: $Wi = 1$, $a = 2$

Tavola degli Accessi

Concetto	Costrutto	Accessi	Tipo
Commento	Relazione	100	R

$$C(Op_2) = 10 \cdot 1 \cdot (2 \cdot 0 + 100) = 1000$$

Costo Dello Schema

$$C(S) = C(Op_1) + C(Op_2) = 20 + 1000 = 1020$$

$$C(S_{\text{rid}}) = C(Op_1) + C(Op_2) = 50 + 10 = 60$$

$$\frac{C(S)}{C(S_{\text{rid}})} = \frac{1020}{60} = 17$$

Occupazione di Memoria

$$M(S) = 277 \cdot 100 + 316 \cdot 50 = 27700 + 15800 = 43500 \text{ byte}$$

$$M(Srid) = 277 \cdot 100 + 320 \cdot 50 = 27700 + 16000 = 43700 \text{ byte}$$

$$M(S) - M(S_{\text{rid}}) = 43500 - 43700 = -200 \text{ byte}$$

In conclusione: Il costo con ridondanza è significativamente inferiore rispetto al costo senza ridondanza. La differenza di utilizzo della memoria è minima, con la ridondanza che utilizza solo 200 byte in più.

2.2 Traduzione nel modello logico

- Utente (IdUtente, Username, Email, Password, N_Seguaci, N_Seguiti)
- Premium (IdUtente*, Tipo*, ScadenzaAbbonamento)
- Abbonamento (Tipo, Tariffa)
- Pagamento (N_Carta, IdUtente*, Intestatario, DataPagamento)
- Blog (IdBlog, IdUtente*, TitoloBlog, Descrizione, Immmagine, N_Follow)
- Coautore(IdCoautore*, IdBlog*)
- FollowUtente (IdUtenteSeguace*, IdUtenteSeguito*)
- FollowBlog(IdUtente*, IdBlog*)
- Categoria (IdCategoria, Tema)
- Sottocategoria (IdCategoria*, IdSottocategoria*)
- Associa (IdBlog*, IdCategoria*)
- Post (IdPost, IdUtente*, IdBlog*, TitoloPost, Testo, Data, N_Like, N_Commenti, N_View)
- Immagine (IdImmagine, IdPost*, Immagine)
- Feedback (IdUtente*, IdPost*, Tipo)
- Commento (IdCommento, IdUtente*, IdPost*, Testo, Data)

3 Data Definition Language

A)
```sql
CREATE TABLE Utente
(
    IdUtente INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(35) UNIQUE NOT NULL,
    Email VARCHAR(50) UNIQUE NOT NULL,
    Password VARCHAR(64) NOT NULL,
    N_Seguaci INT UNSIGNED DEFAULT 0,
    N_Seguiti INT UNSIGNED DEFAULT 0,
    FotoProfilo VARCHAR(255) DEFAULT immagini/profile.png
);
```

B)
```sql
CREATE TABLE Premium
(
    IdUtente INT UNSIGNED PRIMARY KEY,
    Tipo VARCHAR(10),
    ScadenzaAbbonamento DATE NOT NULL,
    FOREIGN KEY (IdUtente) REFERENCES Utente(IdUtente)
    ON DELETE CASCADE,
    FOREIGN KEY (Tipo) REFERENCES Abbonamento(Tipo)
);
```

C)
```sql
 CREATE TABLE Abbonamento
(
  Tipo VARCHAR(10) PRIMARY KEY,
  Tariffa DOUBLE(4,2) NOT NULL
);
```

D)
```sql
CREATE TABLE Pagamento
(
  N_Carta VARCHAR(16) PRIMARY KEY,
  IdUtente INT UNSIGNED,
  Intestatario VARCHAR(70) NOT NULL,
  DataPagamento DATE NOT NULL,
  FOREIGN KEY (IdUtente) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE
);
```

E)
```sql
CREATE TABLE Blog
(
  IdBlog INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  IdUtente INT UNSIGNED,
  TitoloBlog VARCHAR(30) NOT NULL,
  Descrizione VARCHAR(255) NOT NULL,
  Immagine VARCHAR(255) DEFAULT immagini/placeholder.png,
  N_Follow INT UNSIGNED DEFAULT 0,
  FOREIGN KEY (IdUtente) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE
);
```

F)
```sql
CREATE TABLE Coautore
(
  IdCoautore INT UNSIGNED,
  IdBlog INT UNSIGNED,
  PRIMARY KEY(IdCoautore, IdBlog),
  FOREIGN KEY (IdCoautore) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE,
  FOREIGN KEY (IdBlog) REFERENCES Blog(IdBlog)
  ON DELETE CASCADE
);
```

G)
```sql
CREATE TABLE FollowUtente
(
  IdUtenteSeguace INT UNSIGNED,
  IdUtenteSeguito INT UNSIGNED,
  PRIMARY KEY(IdUtenteSeguace, IdUtenteSeguito),
  FOREIGN KEY (IdUtenteSeguace) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE,
  FOREIGN KEY (IdUtenteSeguito) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE
);
```

H)
```sql
CREATE TABLE FollowBlog
(
  IdUtente INT UNSIGNED,
  IdBlog INT UNSIGNED,
  PRIMARY KEY(IdUtente, IdBlog),
  FOREIGN KEY (IdUtente) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE,
  FOREIGN KEY (IdBlog) REFERENCES Blog(IdBlog)
  ON DELETE CASCADE
);
```

I)
```sql
CREATE TABLE Categoria
(
  IdCategoria INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  Tema VARCHAR(50)
);
```

J)
```sql
CREATE TABLE Sottocategoria
(
  IdCategoria INT UNSIGNED,
  IdSottocategoria INT UNSIGNED,
  PRIMARY KEY(IdCategoria, IdSottocategoria),
  Tema VARCHAR(50),
  FOREIGN KEY (IdCategoria) REFERENCES Categoria(IdCategoria),
  FOREIGN KEY (IdSottocategoria) REFERENCES Categoria(IdCategoria)
);
```

K)
```sql
CREATE TABLE Associa
(
  IdBlog INT UNSIGNED,
  IdCategoria INT UNSIGNED,
  PRIMARY KEY(IdBlog, IdCategoria),
  FOREIGN KEY (IdBlog) REFERENCES Blog(IdBlog) ON DELETE CASCADE,
  FOREIGN KEY (IdCategoria) REFERENCES Categoria(IdCategoria)
);
```

L
```sql
CREATE TABLE Post
(
  IdPost INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  IdUtente INT UNSIGNED,
  IdBlog INT UNSIGNED,
  TitoloPost VARCHAR(30) NOT NULL,
  Testo VARCHAR(255) NULL,
  Data DATETIME NOT NULL,
  N_Like INT UNSIGNED DEFAULT 0,
  N_Commenti INT UNSIGNED DEFAULT 0,
  N_View INT UNSIGNED DEFAULT 0,
  FOREIGN KEY (IdUtente) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE,
  FOREIGN KEY (IdBlog) REFERENCES Blog(IdBlog)
  ON DELETE CASCADE
);
```

M
```sql
CREATE TABLE Immagine
(
  IdImmagine INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  IdPost INT UNSIGNED,
  Immagine VARCHAR(255) DEFAULT NULL,
  FOREIGN KEY (IdPost) REFERENCES Post(IdPost)
  ON DELETE CASCADE
);
```

N
```sql
CREATE TABLE Feedback
(
  IdUtente INT UNSIGNED,
  IdPost INT UNSIGNED,
  Tipo TINYINT(1) DEFAULT 0,
  PRIMARY KEY(IdUtente, IdPost),
  FOREIGN KEY (IdUtente) REFERENCES Utente(IdUtente)
  ON DELETE SET NULL,
  FOREIGN KEY (IdPost) REFERENCES Post(IdPost)
  ON DELETE CASCADE
);
```

O
```sql
CREATE TABLE Commento
(
  IdCommento INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  IdUtente INT UNSIGNED,
  IdPost INT UNSIGNED,
  Testo VARCHAR(255) NOT NULL,
  Data DATETIME NOT NULL,
  FOREIGN KEY (IdUtente) REFERENCES Utente(IdUtente)
  ON DELETE CASCADE,
  FOREIGN KEY (IdPost) REFERENCES Post(IdPost)
  ON DELETE CASCADE
);
```

4 Trigger

A) Incrementare like

```
CREATE TRIGGER IncrementaLike
AFTER UPDATE ON Feedback
FOR EACH ROW
BEGIN
  IF NEW.Tipo = 1 THEN
    UPDATE Post
    SET N_Like = N_Like+1
    WHERE IdPost = NEW.IdPost;
  END IF;
END;
```

B) Rimuovere like

```
CREATE TRIGGER RimozioneLike
AFTER UPDATE ON Feedback
FOR EACH ROW
BEGIN
  IF OLD.Tipo = 1 THEN
    UPDATE Post
    SET N_Like = N_Like - 1
    WHERE IdPost = OLD.IdPost AND N_Like > 0;
  END IF;
END;
```

C) Incrementare commenti

```
CREATE TRIGGER IncrementaCommenti
AFTER INSERT ON Commento
FOR EACH ROW
BEGIN
  UPDATE Post
  SET N_Commenti = N_Commenti+1
  WHERE IdPost = NEW.IdPost;
END;
```

D) Rimuovere commenti

```
CREATE TRIGGER RimozioneCommenti
AFTER DELETE ON Commento
FOR EACH ROW
BEGIN
  UPDATE Post
  SET N_Commenti = N_Commenti - 1
  WHERE IdPost = OLD.IdPost AND N_Commenti > 0;
END;
```

E) Incrementare view

```
CREATE TRIGGER IncrementaView
AFTER INSERT ON Feedback
FOR EACH ROW
BEGIN
  UPDATE Post
  SET N_View = N_View+1
  WHERE IdPost = NEW.IdPost;
END;
```

F) Incrementare seguaci

```
CREATE TRIGGER IncrementaSeguaci
AFTER INSERT ON FollowUtente
FOR EACH ROW
BEGIN
  UPDATE Utente
  SET N_Seguaci = N_Seguaci+1
  WHERE IdUtente = NEW.IdUtenteSeguito;
END;
```

G) Rimuovere seguaci

```
CREATE TRIGGER RimozioneSeguaci
AFTER DELETE ON FollowUtente
FOR EACH ROW
BEGIN
  UPDATE Utente
  SET N_Seguaci = N_Seguaci - 1
  WHERE IdUtente = OLD.IdUtenteSeguito AND N_Seguaci > 0;
END;
```

H) Incrementare seguiti

```
CREATE TRIGGER IncrementaSeguiti
AFTER INSERT ON FollowUtente
FOR EACH ROW
BEGIN
  UPDATE Utente
  SET N_Seguiti = N_Seguiti+1
  WHERE IdUtente = NEW.IdUtenteSeguace;
END;
```

I) Rimuovere seguiti

```
CREATE TRIGGER RimozioneSeguiti
AFTER DELETE ON FollowUtente
FOR EACH ROW
BEGIN
  UPDATE Utente
  SET N_Seguiti = N_Seguiti - 1
  WHERE IdUtente = OLD.IdUtenteSeguace AND N_Seguiti > 0;
END;
```

J) Incrementa FollowBlog

```
CREATE TRIGGER IncrementaFollowBlog
AFTER INSERT ON FollowBlog
FOR EACH ROW
BEGIN
  UPDATE Blog
  SET N_Follow = N_Follow+1
  WHERE IdBlog = NEW.IdBlog;
END;
```

K) Rimuovere FollowBlog

```
CREATE TRIGGER RimozioneFollowBlog
AFTER DELETE ON FollowBlog
FOR EACH ROW
BEGIN
  UPDATE Blog
  SET N_Follow = N_Follow - 1
  WHERE IdBlog = OLD.IdBlog AND N_Follow > 0;
END;
```

L) Limite di immagini

```
CREATE TRIGGER NumeroImmagini
BEFORE INSERT ON Immagine
FOR EACH ROW
BEGIN
  DECLARE image_count INT;

  SELECT COUNT(*) INTO image_count
  FROM Immagine
  WHERE IdPost = NEW.IdPost;

  IF image_count >= 5 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Limite di 5 immagini per post raggiunto';
  END IF;
END;
```

M) Limite numero Blog

```
CREATE TRIGGER NumeroBlog
BEFORE INSERT ON Blog
FOR EACH ROW
BEGIN
  DECLARE N_Blog INT;

  SELECT COUNT(*) INTO N_Blog
  FROM Blog
  WHERE IdUtente = NEW.IdUtente;

  IF NEW.IdUtente NOT IN (
    SELECT IdUtente
    FROM Premium
    WHERE ScadenzaAbbonamento > NOW()
    ) AND N_Blog = 5 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = "Non sei autorizzato a creare oltre 5 blog.";
  END IF;
END;
```

5 LoDicoDa

5.1 Funzionalità

5.1.1 Home

La homepage presenta due pulsanti in alto a destra, uno per la registrazione e uno per l'accesso all'area riservata degli utenti.

5.1.2 Homepage

Se il login ha successo, l'utente viene reindirizzato alla homepage. Al primo accesso, è visibile una lista di utenti consigliati, ordinati dal più seguito al meno seguito. Presente un pulsante che consente di visualizzare il profilo degli utenti consigliati. Una volta che l'utente ha seguito altri utenti e blog, nella homepage vengono visualizzati i post dei blog seguiti e i nuovi blog pubblicati dagli utenti seguiti. Possibile cliccare sull'username per visualizzare il profilo del creatore del blog o post. L'header della pagina presenta a sinistra il logo dell'applicazione, che funge da link per tornare alla homepage, mentre a destra si trova un menu a tendina con diverse opzioni di navigazione.

5.1.3 Profilo

Accessibile dal menu a tendina, permette di accedere alla pagina personale dell'utente, e di visualizzare i propri blog tramite una tabella e di visualizzare i blog per i quali l'utente è coautore. Ogni riga della tabella presenta pulsanti per visualizzare o eliminare i blog (per i blog da coautore è presente solo il pulsante per visualizzare).

5.1.4 Blog

Qui è possibile visualizzare il blog specifico e se si è il creatore è possibile anche modificare i dati del blog. Presente anche una tabella con i post pubblicati nel blog e dei pulsanti per visualizzare e, se si è il creatore, eliminare i post.

5.1.5 Post

Qui è possibile visualizzare il post specifico e se si è il creatore è possibile anche modificare i dati del post.

5.1.6 Ricerca

Accessibile dal menu a tendina, è possibile effettuare una ricerca di utenti, blog o post.

5.1.7 Crea Blog

Accessibile dal menu a tendina, pagina per creare un blog, specificando nel form il titolo, la descrizione, la categoria, ed eventualmente una sottocategoria e un'immagine.

5.1.8 Crea Post

Accessibile dal menu a tendina, pagina per creare un post, specificando nel form il blog di appartenenza, il titolo del post, la descrizione e opzionalmente fino a 5 immagini.

5.1.9 Modifica

Accessibile dal menu a tendina, pagina per modificare i dati dell'utente ed eventualmente eliminare il profilo.

5.1.10 Coautore

Accessibile dal menu a tendina, pagina per aggiungere utenti come coautori dei propri blog, tramite un form e per rimuoverne altri attraverso la tabella presente sotto il form.

5.1.11 Sottocategoria

Accessibile dal menu a tendina, pagina per aggiungere ulteriori sottocategorie ai propri blog.

5.1.12 Premium

Accessibile dal menu a tendina, consente di diventare utente premium, scegliendo un abbonamento mensile o annuale, non rinnovabile automaticamente ma richiedente un rinnovo manuale, indicando il numero della carta di credito e l'intestatario.

5.1.13 Logout

Accessibile dal menu a tendina, permette all'utente di terminare la sessione e uscire dall'applicazione web.

5.2 Vincoli

- Ogni utente può creare fino a 5 blog
- Ogni utente può scegliere fino a 3 coautori
- I coautori che un utente può selezionare sono utenti che l'utente ha tra i seguiti
- Ogni utente può scegliere fino a 5 sottocategorie
- Ogni post può avere fino a 5 immagini

5.3 Premium

Per il Pagamento ho adottato il metodo di JQuery Validation Plugin "creditcard" che utilizza la Formula di Luhn, conosciuta anche come Modulo 10, che consente di verificare se il N_Carta è valido o meno.

- Accanto a username, sulla pagina del profilo è possibile visualizzare una stella che attesta il premium
- Numero illimitato di blog
- Numero illimitato di coautori
- Numero illimitato di sottocategorie
- Possibilità di visualizzare i seguiti e i seguaci di un utente, compresi i propri, cliccando su Seguiti e Seguaci
- Possibilità di visualizzare i seguaci di un blog, compresi i propri, cliccando su Seguaci

6 Accessibilità

Per rendere l'applicazione accessibile a tutti gli utenti ho cercato di mantenere uno stile minimale e di essere il più chiaro possibile, ad esempio usando dei pulsanti abbastanza visibili e specificando esattamente la funzionalità: "Modifica", "Elimina", "Visualizza". Ho cercato di fare attenzione anche all'uso dei colori, mantenendo sempre l'uso del verde e del perla ed evitando di usare scritte dello stesso colore del background e infine ho indicato degli alt per ogni foto.

6.1 Tool

Per testare l'accessibilità della mia applicazione web ho usato Silktide, uno strumento molto utile che mi ha permesso di simulare diversi tipi di disabilità e diversi tipi di dispositivi.

Figure 1: Test sui colori

Nel secondo test, si può notare che Silktide mi fa notare che in alcune pagine ci sono diversi pulsanti che rimandano a pagine differenti, ma dallo screen è anche possibile notare che in ogni riquadro ho specificato se si tratta di un post, di un utente o di un blog, per rendere chiaro ciò.

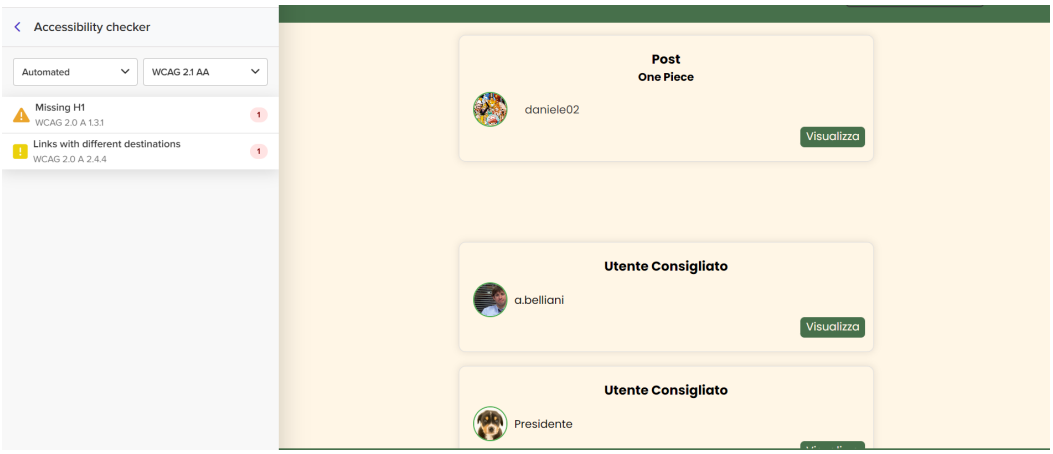


Figure 2: Test accessibilità

Nel terzo test, ho provato a simulare il funzionamento dell'app sui vari dispositivi. L'app risulta accessibile da pc e quasi interamente su tablet, meno su dispositivi mobile. Questo potrebbe essere risolto attraverso l'uso di misure relative (ad esempio le percentuali) e non fisse (come px).

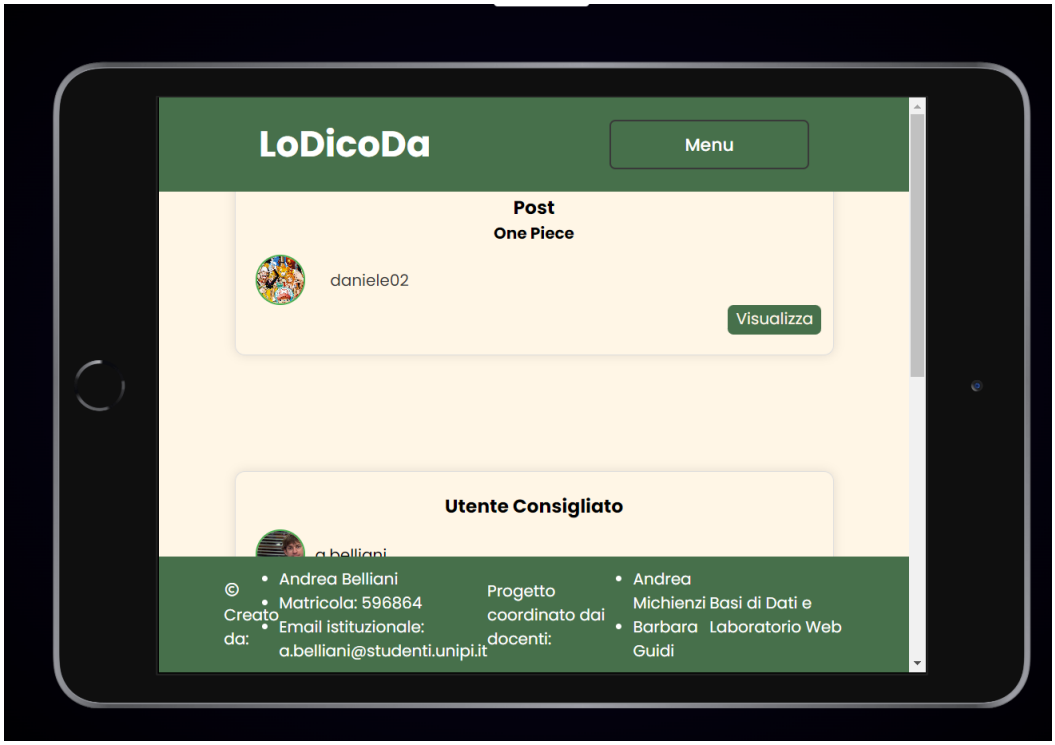


Figure 3: Test dispositivi