

RELAZIONE del PROGETTO di
BASI DI DATI
SIMONA SETTE
Matricola 544298

INDICE:

- Pagina 3→Specifiche di sistema
- Pagina 5→Modello entità-relazione
- Pagina 6→Dizionario entità
- Pagina 7→Dizionario relazioni
- Pagina 8→Traduzione delle relazioni
- Pagina 10→Analisi delle prestazioni del modello E-R
- Pagina 16→Modello relazionale e vincoli di integrità
- Pagina 17→Database SQL
- Pagina 27→Tabella delle operazioni
- Pagina 28→Tabella business rules
- Pagina 29→Guida alla cartella *thoughts*
- Pagina 30→Scelte di progetto
- Pagina 31→Scelte per i controlli degli input
- Pagina 35→ Screenshot delle pagine
- Pagina 51→Documentazioni funzioni php/metodi jquery e elementi bootstrap utilizzati

Specifiche di sistema

Si vuole progettare una base di dati per un sistema collaborativo on-line adibito ad ospitare blog realizzati da utenti registrati al sistema.

Esistono due tipi di utenti: l'utente registrato e l'utente visitatore.

L'utente visitatore è un utente che non ha effettuato il login oppure un utente che non è registrato nel sistema. Questo utente può visualizzare la homepage nella sua interezza, utilizzare le funzioni di ricerca dei blog, visitare i blog ed effettuare un'iscrizione al sito.

Un utente registrato al sistema è un utente di cui noi conosciamo il nome, cognome, età, email, password, sesso, numero di telefono, numero di documento di identità e il nickname. Il numero di telefono, il numero di documento e il nickname sono unici per ogni utente.

Un utente registrato può modificare i propri dati, la propria password, cancellare i propri commenti su qualsiasi post, creare un massimo di 5 blog e cancellare i propri blog, seguire un blog di suo interesse, creare e cancellare i propri post, mettere e togliere like da qualsiasi post, modificare il proprio blog, nominare/rimuovere un altro registrato come collaboratore dei propri blog, cancellare tutti i commenti sui propri blog, cancellare i propri commenti su post altrui ed eliminare il proprio account.

Quando un account viene eliminato, se l'utente ha dei blog e questi blog hanno dei collaboratori, l'utente può decidere se lasciare quei blog in eredità ai suoi collaboratori oppure rimuoverli dalla collaborazione ed eliminare i propri blog insieme a sé stesso. Per questo stesso meccanismo, può decidere di nominare collaboratori su blog in cui non ne aveva, in modo che i propri blog sopravvivano alla sua cancellazione di account.

Tolto il caso particolare sopra esplicitato, se un utente registrato cancella il proprio account perde i diritti da utente registrato e tutte le azioni da lui compiute e cose da lui realizzate sul sito vengono rimosse dal sistema insieme a lui.

Un utente **non** può mettere il "segui" sui propri blog o sui blog con cui collabora. Se un utente segue un blog e in un secondo momento venisse nominato collaboratore, quel segui viene rimosso.

Ogni blog ha un codice univoco, un titolo, un creatore, un eventuale collaboratore, un colore di sfondo e uno di font. Ogni Blog ha un tema, eventuali/e sottotemi/a, tutti singolarmente identificati da un codice univoco.

I colori di sfondo e font sono predefiniti e possono essere scelti liberamente tra le opzioni a disposizione; sia i colori di sfondo che i colori di font sono identificati da un codice univoco.

Il ruolo di collaboratore deve essere assegnato (e può essere rimosso) dal proprietario del blog e deve essere un utente registrato nel sistema. Il collaboratore può decidere di smettere di assumere quel ruolo per uno specifico blog e tornare un semplice utente.

Un collaboratore può fare tutto quello che può fare l'autore, tranne eliminare il blog su cui collabora; può, inoltre, ereditare la proprietà del blog su cui collabora (diventando il nuovo autore) se il proprietario corrente cancella il proprio account e decide di lasciare i blog in eredità, oppure se il proprietario vuole rinunciare alla proprietà del proprio blog.

Un blog può contenere post o essere vuoto.

Ogni post è identificato da un codice univoco ed è composto da titolo, testo, data e ora di inserimento e un massimo di tre immagini; è inoltre presente, per ogni post, il codice identificativo del blog su cui sono stati scritti.

I post possono essere generati e cancellati solo dall'autore o dal collaboratore del blog.

Ogni immagine dispone di un codice univoco ed è caratterizzato dal path dove è localizzata l'immagine e da un riferimento al post a cui sono allegate.

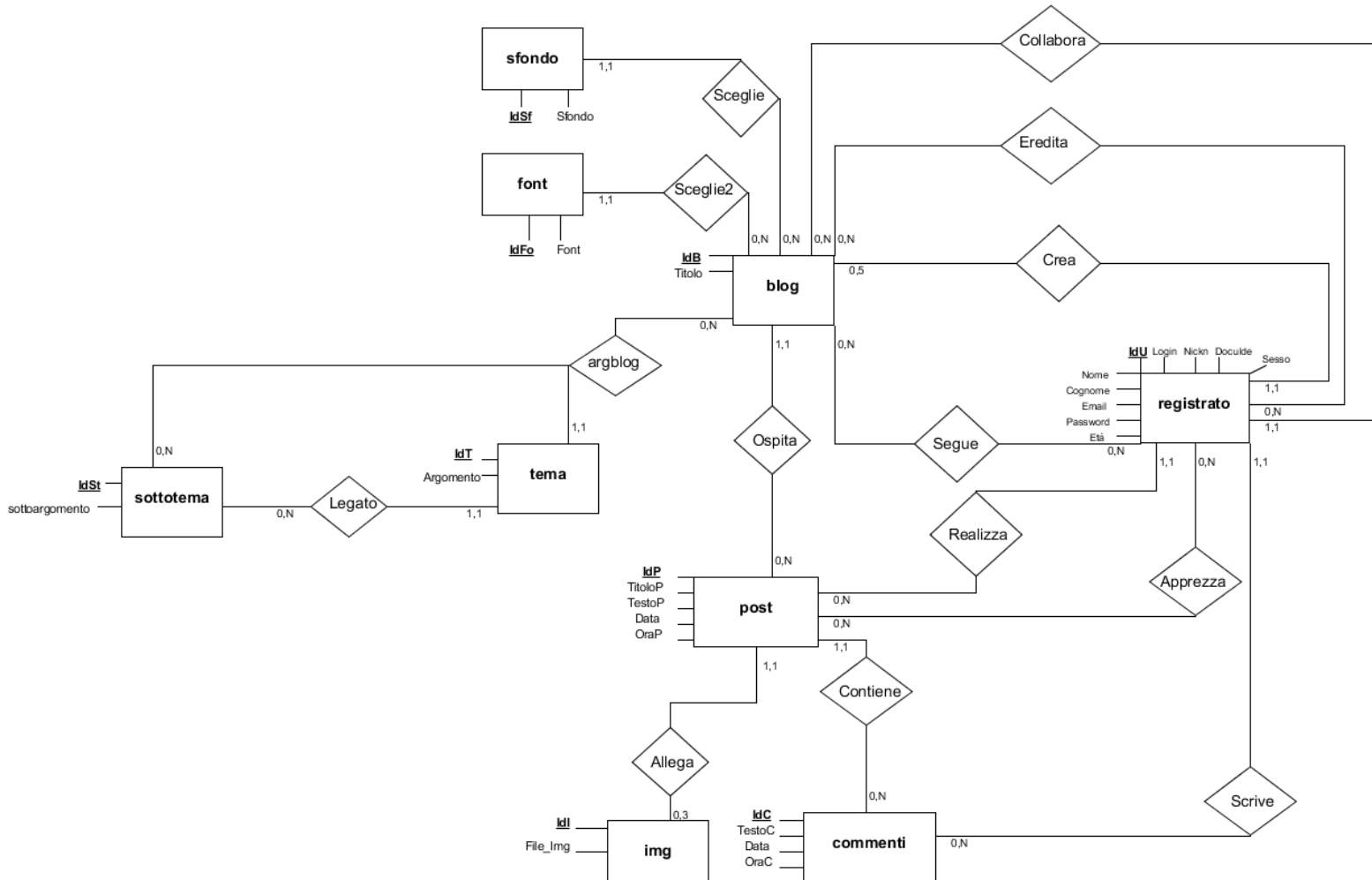
Ogni post ammette commenti.

I commenti sono identificati da un codice univoco e sono caratterizzati da un riferimento al proprietario del commento e al post su cui sono stati creati, un testo, data e ora di creazione.

I commenti possono essere rimossi solo dal proprietario del blog, dall'eventuale collaboratore del blog e dall'autore del commento.

Ogni immagine dispone di un codice univoco.

Modello entita'-relazione



Dizionario delle entità

| Entità | Descrizione | Attributi | Identificatore |
|------------|---|---|----------------|
| blog | Blog realizzato da un utente registrato | IdU, IdSf, IdFo, IdUcollab, Titolo, Ereditato | idB |
| registrato | Un utente registrato può interagire, visualizzare e creare un blog, visualizzare, commentare e mettere mi piace ad un post. | Nome, Cognome, Eta, Email, Password, Sesso, Tel, DocuIde, NickN | idU |
| sfondo | Posseduto dai blog | sfondo | IdSf |
| font | Posseduto dai blog | font | IdFo |
| tema | Argomento di cui tratta il Blog | Argomento | idT |
| sottotema | Sottoargomenti specifici dei singoli temi, che possono essere posseduti dai blog. | Sottoargomento, IdT | IdSt |
| post | Contenuti pubblicati all'interno di un blog da un utente registrato | IdB, TitoloP, TestoP, DataP, OraP | IdP |

| | | | |
|----------|---|-------------------------------|-----|
| Img | Immagini allegabili ad un post | Idp, File_Img | IdI |
| Commenti | Contenuti testuali legati ad un post, inseriti da utenti registrati | Idp, IdU, TestoC, DataC, OraC | idC |

Dizionario delle relazioni

| Relazione | Descrizione | Tipo |
|-----------|--------------------------------------|--------------|
| Allega | Lega le entità Post e Img | Uno a Tre |
| Contiene | Lega le entità Post e Commenti | Uno a molti |
| Scrive | Lega le entità Commenti e Registrato | Uno a molti |
| Ospita | Lega le entità Blog e Post | Uno a molti |
| Realizza | Lega le entità Post e Registrato | Uno a molti |
| Crea | Lega le entità Blog e Registrato | Uno a Cinque |
| Collabora | Lega le entità Blog e Registrato | Uno a Molti |

| | | |
|----------|---|---|
| Sceglie | Lega le entità Blog e Sfondo | Uno a molti |
| Sceglie2 | Lega le entità Blog e Font | Uno a molti |
| Legato | Lega l'entità Tema con l'entità sottotema | Uno a molti |
| Apprezza | Lega le entità Post e Registrato | Molti a Molti |
| argblog | Lega l'entità blog con le entità tema e sottotema | <ul style="list-style-type: none"> • Blog-tema: Uno a molti • Blog-sottotema: Molti a molti |
| eredita | Lega le entità registrato e blog | Molti a molti |
| segue | Lega le entità blog e registrato | Molti a molti |

Traduzione delle relazioni:

Relazioni uno a molti: ho optato per la possibilità di inglobare la relazione nell'entità con cardinalità massima 1

- ***Allega (rel. 1 a 3):** reso tramite foreign key dentro la tabella img; la traduzione qui ha seguito il modello ideale di traduzione delle relazioni m-m perché se $\text{min-card}(\text{img}, R) = 0$ e $\text{min-card}(\text{post}, R) = 1$ la chiave derivante da **img** ammetterà valori nulli, e la chiave primaria si ottiene da **post(Foreign key nella tabella img)**
- **Contiene:** reso tramite foreign key dentro la tabella commenti
- **Scrive:** reso tramite foreign key all'interno della tabella commenti
- **Ospita:** reso tramite foreign key all'interno della tabella post

- **Realizza**: reso tramite il fatto che **solo** il collaboratore e il proprietario di un blog possono creare dei post su di esso. La **proprietà** dei post non è legata agli autori, ma al blog: questo permette la completa e libera dinamicità dei ruoli di potere all'interno di un blog
- **Crea**: reso tramite foreign key all'interno della tabella blog
- **Collabora**: reso tramite attributo IdUcollab all'interno della tabella blog, che fa riferimento alla primary key della tabella registrato
- **Sceglie** e **sceglie2**: rese tramite foreign key all'interno della tabella blog
- **Legato**: reso tramite foreign key all'interno della tabella sottotema

Traduzione delle relazioni restanti (molti a molti):

- **Apprezza**→reso tramite una tabella a sé, contenente gli attributi IdU, IdP, foreign key che fanno riferimento alle primary key delle entità Post e Registrato
- **argblog**→reso tramite una tabella a sé, contenente gli attributi IdT, IdSt, IdB, foreign key che fanno riferimento alle primary key delle entità tema, sottotema e blog
- **eredita**→reso tramite una tabella a sé, contenente gli attributi IdB, IdU, foreign key che fanno riferimento alle primary key delle entità Registrato e Blog
- **segue**→reso tramite una tabella a sé, contenente gli attributi IdB, IdU, foreign key che fanno riferimento alle primary key delle entità Blog e Registrato

Analisi delle prestazioni del modello E-R

TABELLA DEI VOLUMI

Nei volumi inserisco una stima **del numero medio di occorrenze** delle entità/relazione in questione.

Considerato che per un calcolo della media preciso sarebbe necessario avere dati precisi (la somma dei valori numerici divisa per il numero di valori numerici considerati), i numeri inseriti nella tabella dei volumi **sono stime assunte**, non rispecchiano l'effettivo volume nel database consegnato.

| Concetto | Tipo | Volume |
|---------------------|------|---|
| registrato (utente) | E | 100 |
| blog | E | Assumo che un registrato abbia in media 2 blog- 200 |
| post | E | Illimitati per ogni blog-assumo che un blog abbia in media 2 post- 400 |
| commenti | E | Illimitati per ogni post-assumo che un post abbia in media 2 commenti- 800 |
| Img (immagini) | E | assumo che un post abbia in media 2 foto- 800 |
| tema | E | 20 argomenti disponibili |
| sottotema | E | Sotto argomenti variabili a seconda dell'argomento- 81 |
| sfondo | E | 8 sfondi disponibili |
| font | E | 8 colori font disponibili |
| apprezza | R | (opzionale)1 per post- 400 |
| segue | R | (opzionale)1 per blog- 200 |
| collabora | R | (opzionale)1 per blog- 200 |
| argblog | R | <ul style="list-style-type: none">1 tema per blog-200Sottotemi Variabili a seconda dell'argomento e opzionali- assumo che un blog abbia in media 1 sottotema-200$200+200=$400 |
| Sceglie | R | 1 sfondo per blog- 200 |
| Sceglie2 | R | 1 font per blog- 200 |
| Legato | R | $81/20=4,05$; un tema ha in media 4,05 sottotemi- 81 (corrisponde al numero totale dei sottotemi) |

| | | |
|----------|---|---|
| Eredita | R | Assumo che un utente erediti in media 1 blog- 200 |
| Crea | R | Assumo che un registrato crei in media 2 blog- 200 |
| Ospita | R | assumo che un blog abbia in media 2 post- 400 |
| Realizza | R | Assumo che un registrato crei in media 4 post- 400 |
| Scriva | R | Assumo che un registrato crei in media 4 commenti- 400 |
| Contiene | R | assumo che un post abbia in media 2 commenti- 800 |
| allega | R | Massimo 3 per post -assumo che un post abbia in media 2 foto- 800 |

Data un'operazione O di tipo T valutiamo il suo costo $c(O_t)$ come:

$$C(O_r) = f(O_r) * wp * (\alpha * NC_w + NC_r)$$

La frequenza dell'operazione che si moltiplica per il peso dell'operazione , il tutto moltiplicato per il coefficiente moltiplicativo delle operazioni in scrittura moltiplicato al numero di accessi in scrittura (se vi sono) che si somma al numero di accessi in lettura.

Supponiamo:

- Peso dell'operazione (interattiva o batch) **wp = 0,5**
- Coefficiente moltiplicativo delle operazioni in scrittura **$\alpha = 2$**

Operazioni di lettura

- Operazione 1 → Visualizzare i blog di un utente specifico.
- Operazione 2 → Visualizzare i post di un blog specifico.
- Operazione 3 → Dato l'identificativo dell'utente registrato, visualizzare i dati di questo.

| Operazione | Tipo | Frequenza |
|--------------|------|-----------------|
| Operazione 1 | I | 50 volte/giorno |
| Operazione 2 | I | 30 volte/giorno |
| Operazione 3 | I | 50 volte/giorno |

Tavole degli accessi: descrivono (in maniera astratta) il costo di utilizzo dello schema da parte della operazione, inteso come tempo speso dalla operazione.

Tavola accessi per Operazione 1

| Concetto | Costrutto | Accessi | Tipo |
|------------|-----------|---------|------|
| registrato | E | 1 | R |
| crea | R | 1 | R |
| blog | E | 2 | R |

**2 accessi perché in media un registrato possiede due blog*

$$C(O_1) = 50 * 0,5 * (1 + 1 + 2) = 100$$

Tavola accessi per Operazione 2

| Concetto | Costrutto | Accessi | Tipo |
|----------|-----------|---------|------|
| blog | E | 1 | R |
| ospita | R | 1 | R |
| post | E | 2 | R |

**2 accessi perché in media un blog possiede due post*

$$C(O_2) = 30 * 0,5 * (1 + 2 + 1) = 60$$

Tavola accessi per Operazione 3

| Concetto | Costrutto | Accessi | Tipo |
|------------|-----------|---------|------|
| registrato | E | 1 | R |

$$C(O_3) = 50 * 0,5 * (1) = 25$$

Operazioni di scrittura

- Operazione 4 → Dato l'identificativo del post, il registrato mette un like ad un post
- Operazione 5 → Dato l'identificativo del blog, il registrato comincia a seguire un blog
- Operazione 6 → Dato l'identificativo dell'utente registrato, nominare questo come collaboratore su un proprio blog
- Operazione 7 → Iscrizione di un nuovo utente al sito.
- Operazione 8 → Dato l'identificativo dell'utente registrato, il registrato crea un nuovo blog

| Operazione | Tipo | Frequenza |
|--------------|------|-----------------|
| Operazione 4 | I | 15 volte/giorno |

| | | |
|--------------|---|----------------|
| Operazione 5 | I | 3 volte/giorno |
| Operazione 6 | I | 4 volte/giorno |
| Operazione 7 | I | 3 volte/giorno |
| Operazione 8 | I | 2 volte/giorno |

Tavola accessi per Operazione 4

| Concetto | Costrutto | Accessi | Tipo |
|------------|-----------|---------|------|
| registrato | E | 1 | R |
| apprezza | R | 1 | W |

- *Lettura dati su registrato*
- *Scrittura dati su apprezza*

$$C(O_4)=15*0,5*(2*1+1)=22,5$$

Tavola accessi per Operazione 5

| Concetto | Costrutto | Accessi | Tipo |
|------------|-----------|---------|------|
| registrato | E | 1 | R |
| segue | R | 1 | W |

- *Lettura dati su registrato*
- *Scrittura dati su segue*

$$C(O_5)=3*0,5*(2*1+1)=4,5$$

Tavola accessi per Operazione 6

| Concetto | Costrutto | Accessi | Tipo |
|-----------|-----------|---------|------|
| blog | E | 1 | R |
| collabora | R | 1 | W |

- *Lettura dati su blog*
- *Scrittura dati su collabora*

$$C(O_6)=4*0,5*(2*1+1)=6$$

Tavola accessi per Operazione 7

| Concetto | Costrutto | Accessi | Tipo |
|------------|-----------|---------|------|
| registrato | E | 1 | W |

- Scrittura dati su registrato

$$C(O_7) = 3 * 0,5 * (2 * 1) = 3$$

Tavola accessi per Operazione 8

| Concetto | Costrutto | Accessi | Tipo |
|----------|-----------|---------|------|
| blog | E | 1 | W |
| crea | R | 1 | W |

- Scrittura dati su blog
- Aggiornamento dati su crea

$$C(O_8) = 2 * 0,5 * (2 * 2) = 4$$

Supponendo queste operazioni sui dati come l'insieme totale di operazioni sui dati $0_1, 0_2 \dots 0_8$ con costi $c(0_1), c(0_2) \dots c(0_8)$, il **costo dello schema** è definito come la somma dei costi di tutte le operazioni fatte sullo schema E-R:

$$c(S) = 100 + 60 + 25 + 22,5 + 4,5 + 6 + 3 + 4 = \mathbf{225 \text{ costo delle operazioni in un giorno}}$$

Occupazione di memoria dello schema è dato dalla somma di occupazione di memoria delle entità e occupazione di memoria delle relazioni

- assumo che ogni attributo abbia dimensione di 6 byte per semplificare i calcoli

$$\text{Memoria occupata (registrato)} = 100 * (10 \text{ attributi} * 6 \text{ byte}) = 6000$$

$$\text{Memoria occupata (blog)} = 200 * (7 \text{ attributi} * 6 \text{ byte}) = 8400$$

$$\text{Memoria occupata (post)} = 400 * (6 \text{ attributi} * 6 \text{ byte}) = 14400$$

$$\text{Memoria occupata (img)} = 800 * (3 \text{ attributi} * 6 \text{ byte}) = 14400$$

$$\text{Memoria occupata (commenti)} = 800 * (6 \text{ attributi} * 6 \text{ byte}) = 28800$$

$$\text{Memoria occupata (tema)} = 200 * (2 \text{ attributi} * 6 \text{ byte}) = 2400$$

$$\text{Memoria occupata (sottotema)} = 200 * (3 \text{ attributi} * 6 \text{ byte}) = 3600$$

$$\text{Memoria occupata (sfondo)} = 200 * (2 \text{ attributi} * 6 \text{ byte}) = 2400$$

$$\text{Memoria occupata (font)} = 200 * (2 \text{ attributi} * 6 \text{ byte}) = 2400$$

Memoria occupata (apprezza)=400(2attributi*6byte)=4800*

Memoria occupata (segue)=200(2attributi*6byte)=2400*

Memoria occupata (collabora)=200(1attributo*6byte)=1200*

Memoria occupata (argblog)=400(3attributi*6byte)=7200*

Memoria occupata (sceglie)=200(1attributi*6byte)=1200*

Memoria occupata (sceglie2)=200(1attributi*6byte)=1200*

Memoria occupata (legato)=81(1attributi*6byte)=486*

Memoria occupata (eredita)=200(1attributi*6byte)=1200*

Memoria occupata (crea)=200(1attributi*6byte)=1200*

Memoria occupata (ospita)=400(1attributi*6byte)=2400*

Memoria occupata (realizza)=400(0attributi*6byte)=4800*

Memoria occupata (scrive)=400(1attributi*6byte)=2400*

Memoria occupata (contiene)=800(1attributi*6byte)=4800*

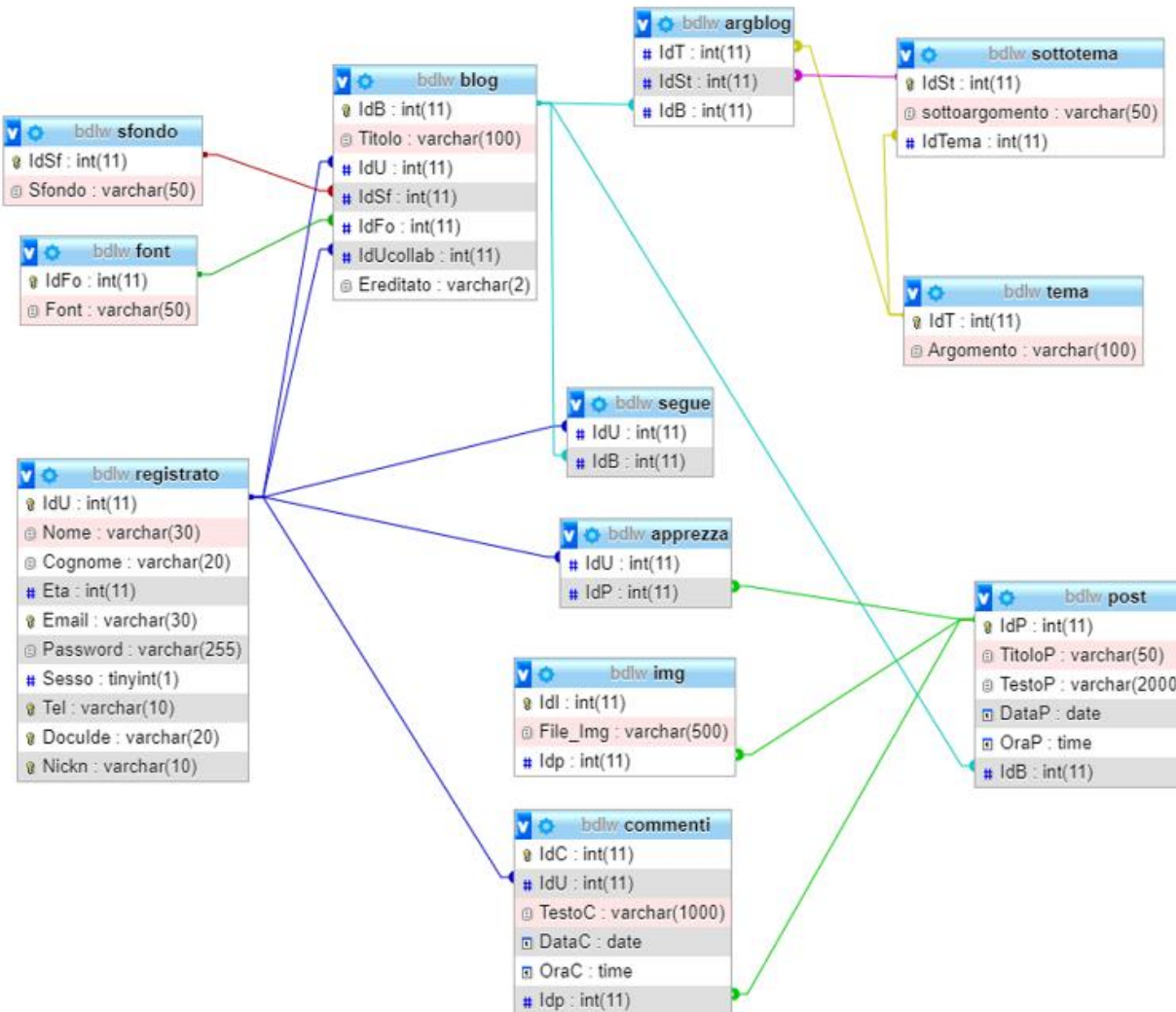
Memoria occupata (allega)=800(1attributi*6byte)=4800*

Memoria occupata dallo schema:

*6000+8400+28800+3600+7200+486+400+(14400*2)+(2400*6)+(4800*3)+(1200*5)=**118.486 byte***

Modello relazionale e vincoli di integrità

VISTE



| bdlw post_like |
|--------------------------|
| # Post : int(11) |
| # TitoloP : varchar(50) |
| # Testo : varchar(2000) |
| # Blog : int(11) |
| # TitoloB : varchar(100) |
| # IDProp : int(11) |
| # IDCollab : int(11) |
| # nLike : bigint(21) |

| bdlw argblogview |
|-------------------------|
| # IdT : int(11) |
| # IdSt : int(11) |
| # IdB : int(11) |
| # Titolo : varchar(100) |
| # IdU : int(11) |
| # IdSf : int(11) |

| bdlw blog_nick |
|--------------------------|
| # IdB : int(11) |
| # Titolo : varchar(100) |
| # IdU : int(11) |
| # Nickn : varchar(10) |
| # IdSf : int(11) |
| # IdFo : int(11) |
| # IdUcollab : int(11) |
| # Ereditato : varchar(2) |

1. Registrato (**IdU**, Nome, Cognome, Eta, Email, Password, Sesso, Tel, DocuIde, NickN)
2. Blog (**IdB**, IdU, IdSf, IdFo, IdUcollab, Titolo, Ereditato)
3. Sfondi (**IdSf**, Sfondi)
4. Fonti (**IdFo**, Fonti)
5. Tema (**IdT**, Argomento)
6. Sottotema (**IdSt**, sottoargomento, IdTema)
7. Post (**IdP**, IdB, TitoloP, TestoP, DataP, OraP)
8. Immagini (**IdI**, Idp, File_Img)
9. Commenti (**IdC**, Idp, IdU, TestoC, DataC, OraC)
10. Apprezza (IdU, IdP)
11. Segue (IdU, IdB)
12. Argblog (IdB, IdSt, IdT)

Database: SQL

- Nome del database: bdlw
- Codifica del database: utf8mb4_general_ci

POST

```
CREATE TABLE post(  
  IdP int AUTO_INCREMENT PRIMARY KEY,  
  TitoloP varchar(50) not null,  
  TestoP varchar(2000) not null,  
  DataP date not null,  
  OraP time not null,  
  IdB int not null,  
  FOREIGN KEY(IdB) REFERENCES blog(IdB)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE);
```

Scelte sui vincoli di integrità della tabella:

- IdB → Quando un blog viene cancellato/modificato sulla tabella madre, la stessa riga che contiene il riferimento a tale dato sulla tabella figlia viene aggiornato/eliminato (ergo tutti i post del blog cancellato vengono cancellati)

IMG

```
CREATE TABLE img(  
  IdI int AUTO_INCREMENT PRIMARY KEY,  
  File_Img varchar(500) not null,  
  Idp int not null,  
  FOREIGN KEY (Idp) REFERENCES post(IdP)  
  On DELETE CASCADE  
  ON UPDATE CASCADE);
```

Scelte sulla tabella:

- L'attributo File_Img ha una dimensione elevata (500 caratteri) per via del fatto che contiene il path all'immagine.

Scelte sui vincoli di integrità della tabella:

- Idp → Quando un post viene cancellato/modificato sulla tabella madre, la stessa riga che contiene il riferimento a tale dato sulla tabella figlia viene aggiornato/eliminato (ergo tutte le immagini del post che viene eliminato vengono eliminate di conseguenza)

COMMENTI

```
CREATE TABLE commenti(
    IdC int AUTO_INCREMENT PRIMARY KEY,
    IdU int not null,
    TestoC varchar(1000) not null,
    DataC date not null,
    OraC time not null,
    Idp int not null,
    FOREIGN KEY (Idp) REFERENCES post(IdP)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (IdU) REFERENCES registrato(IdU)
        ON DELETE CASCADE
        ON UPDATE CASCADE);
```

Scelte sui vincoli di integrità della tabella:

- Idp → Quando un post viene cancellato/modificato sulla tabella madre, la stessa riga che contiene il riferimento a tale dato sulla tabella figlia viene aggiornato/eliminato (ergo tutti i commenti del post che viene eliminato vengono eliminati di conseguenza)
- IdU → Quando l'utente che ha scritto il commento viene cancellato/modificato sulla tabella madre, la stessa riga che contiene il riferimento a tale dato sulla tabella figlia viene aggiornato/eliminato (ergo tutti i commenti dell'utente che viene eliminato vengono eliminati di conseguenza)

TEMA

```
CREATE TABLE tema(  
  IdT int AUTO_INCREMENT PRIMARY KEY,  
  Argomento varchar(100) not null);
```

SOTTOTEMA

```
CREATE TABLE sottotema(  
  IdSt int AUTO_INCREMENT PRIMARY KEY,  
  sottoargomento varchar(50) not null,  
  IdTema int not null,  
  FOREIGN KEY (IdTema) REFERENCES tema(IdT)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE);
```

Scelte sulla tabella:

- L'attributo IdTema è not null perché ogni sottoargomento, per esistere, deve essere subordinato ad un macro-argomento (tema.IdT)

Scelte sui vincoli di integrità della tabella:

- IdTema → Quando un argomento viene cancellato/modificato sulla tabella madre (tema), la stessa riga che contiene il riferimento a tale dato sulla tabella figlia viene aggiornato/eliminato (ergo tutti sottoargomenti di un argomento che viene cancellato, vengono cancellati)

ARGBLOG

```
CREATE TABLE argblog(  
  IdT int not null,  
  IdSt int,  
  IdB int not null,  
  FOREIGN KEY (IdT) REFERENCES tema(IdT)  
  ON DELETE RESTRICT  
  ON UPDATE RESTRICT,  
  FOREIGN KEY (IdSt) REFERENCES sottotema(IdSt)
```

ON DELETE **RESTRICT**
ON UPDATE **RESTRICT**,
FOREIGN KEY (IdB) REFERENCES blog(IdB)
ON DELETE **CASCADE**
ON UPDATE **CASCADE**);

Scelte sulla tabella:

- L'attributo IdSt ammette valori null in quanto le scelte di sottoargomenti per un blog sono opzionali;

Scelte sui vincoli di integrità della tabella:

- IdB → Quando un blog viene cancellato/modificato sulla tabella madre (tema), la stessa riga che contiene il riferimento a tale dato sulla tabella figlia viene aggiornato/eliminato (tutte le righe che contengono l'id di tale blog vengono cancellate)
- IdT → Nel caso ci fosse un tentativo da parte dell'amministratore del database di modificare/eliminare un argomento dalla tabella madre(tema) ma questo argomento esiste all'interno della tabella figlia (argblog in questo caso), il tentativo di modifica/eliminazione fallisce;
La scelta di utilizzare **restrict** deriva dalla mia scelta personale di mettere in primo piano il bisogno del cliente e tenere conto della responsabilità, di chi mette a disposizione un servizio ad un cliente, di non tirarsi indietro: **se ho reso disponibile una feature e un cliente l'ha scelta, ho il dovere di mantenerla disponibile fino a quando sarà in uso dai clienti.**
- IdSt → Nel caso ci fosse un tentativo da parte dell'amministratore del database di modificare/eliminare un sottoargomento dalla tabella madre(sottotema) ma questo sottoargomento esiste all'interno della tabella figlia (argblog in questo caso), il tentativo di modifica/eliminazione fallisce;
La scelta di utilizzare **restrict** deriva dalla mia scelta personale di mettere in primo piano il bisogno del cliente e tenere conto della responsabilità, di chi mette a disposizione un servizio ad un cliente, di non tirarsi indietro: **se ho reso disponibile una feature e un cliente l'ha scelta, ho il dovere di mantenerla disponibile fino a quando sarà in uso dai clienti.**

SFONDO

```
CREATE TABLE sfondo(  
  IdSf int AUTO_INCREMENT PRIMARY KEY,  
  Sfondo varchar(50) not null);
```

FONT

```
CREATE TABLE font(  
  IdFo int AUTO_INCREMENT PRIMARY KEY,  
  Font varchar(50) not null);
```

BLOG

```
CREATE TABLE blog(  
  IdB int AUTO_INCREMENT PRIMARY KEY,  
  Titolo varchar(100) not null,  
  IdU int not null,  
  IdSf int not null,  
  IdFo int not null,  
  IdUcollab int,  
  Ereditato varchar(2),  
  FOREIGN KEY (IdU) REFERENCES registrato(IdU)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
  FOREIGN KEY (IdUcollab) REFERENCES registrato(IdU)  
    ON DELETE SET NULL  
    ON UPDATE CASCADE,  
  FOREIGN KEY (IdSf) REFERENCES sfondo(IdSf)  
    ON DELETE RESTRICT  
    ON UPDATE RESTRICT,  
  FOREIGN KEY (IdFo) REFERENCES font(IdFo)  
    ON DELETE RESTRICT  
    ON UPDATE RESTRICT);
```

Scelte sulla tabella:

- L'attributo Ereditato ammette valori null in quanto non tutti i blog sono stati dati in eredità: i blog la cui proprietà (IdU) corrisponde a quella iniziale (di quando è stato creato il blog) avranno il valore null per questo attributo.
- L'attributo IdUcollab ammette valori null in quanto non tutti i blog hanno un utente che collabora nella loro gestione.

Scelte sui vincoli di integrità della tabella:

- IdU → Quando un utente viene cancellato/modificato sulla tabella madre (registrato), la riga che contiene il riferimento a tale dato sulla tabella figlia viene aggiornato/eliminato (Se l'utente cancella il proprio account dal sito, tutti i blog di cui lui è proprietario (tranne nel caso esista un collaboratore per tale blog, in tal caso i blog vengono ereditati da questo e di conseguenza non saranno cancellati) saranno eliminati.
- IdUcollab → Se c'è una modifica dell'IdU (nella tabella madre "registrato") dell'utente che ha ruolo di collaboratore per un determinato blog, il valore che si riferisce ad esso nella tabella figlia viene modificato di conseguenza;

Se c'è un'eliminazione dell'IdU nella tabella madre (eliminazione dell'utente in registrato), il valore che si riferisce ad esso nella tabella figlia viene settato a null;

- IdSf e IdFo → Nel caso ci fosse un tentativo da parte dell'amministratore del database di modificare/eliminare un colore di font o un colore di sfondo dalle tabelle madri (font, sfondo) ma questo colore di font o di sfondo esiste all'interno della tabella figlia (blog in questo caso), il tentativo di modifica/eliminazione fallisce;
La scelta di utilizzare **restrict** deriva dalla mia scelta personale di mettere in primo piano il bisogno del cliente e tenere conto della responsabilità, di chi mette a disposizione un servizio ad un cliente, di non tirarsi indietro: **se ho reso disponibile una feature e un cliente l'ha scelta, ho il dovere di mantenerla disponibile fino a quando sarà in uso dai clienti.**

REGISTRATO

```
CREATE TABLE registrato(  
  IdU int AUTO_INCREMENT PRIMARY KEY,
```

Nome varchar(30) not null,
Cognome varchar(20) not null,
Eta int not null,
Email varchar(30) not null **UNIQUE**,
Password varchar(255) not null,
Sesso tinyint(1) not null,
Tel varchar(10) not null **UNIQUE**,
DocuIde varchar(20)not null **UNIQUE**,
Nickn varchar(10) not null **UNIQUE**);

Nota: mysql converte boolean in tinyint(1) automaticamente (attributo "Sesso")

Scelte sulla tabella:

- Email, numero di telefono, il codice identificativo del documento d'identità e il nickname sono unici per ogni registrato;
- Motivazione del varchar(225) dell'attributo Password: all'interno del database vengono memorizzate le password codificate da questa funzione di hashing: **password_hash(\$string, PASSWORD_DEFAULT)**;

Fonte: <https://www.php.net/manual/en/function.password-hash.php>

Estratto (tradotto) dalla fonte: **PASSWORD_DEFAULT** - Utilizza l'algoritmo bcrypt (predefinito a partire da PHP 5.5.0).

Nota che questa costante è progettata per cambiare nel tempo man mano che nuovi e più potenti algoritmi vengono aggiunti a PHP. Per questo motivo, la lunghezza del risultato derivante dall'utilizzo di questo identificatore può cambiare nel tempo. Pertanto, si consiglia di memorizzare il risultato in una colonna del database che può espandersi oltre i 60 caratteri (255 caratteri sarebbero una buona scelta)."

Per questo motivo il campo Password all'interno della tabella "registrato" ha la lunghezza massima di 255 caratteri.

SEGUE

CREATE TABLE segue(

```
IdU int not null,  
IdB int not null,  
FOREIGN KEY (IdU) REFERENCES registrato(IdU)  
ON DELETE CASCADE  
ON UPDATE CASCADE,  
FOREIGN KEY (IdB) REFERENCES blog(IdB)  
ON DELETE CASCADE  
ON UPDATE CASCADE);
```

Scelte sui vincoli di integrità della tabella:

-Entrambe le foreign key hanno cascade sia in update che delete perché sia nel caso che cessi di esistere il registrato che ha "seguito" la pagina, sia che cessi di esistere il blog che viene seguito da x utenti/e, voglio che la/le righe a cui fanno riferimento i rispettivi valori nella tabella figlia vengano eliminati (stesso discorso per l'update).

APPREZZA

```
CREATE TABLE Apprezza(  
IdU int not null,  
IdP int not null,  
FOREIGN KEY (IdP) REFERENCES post(IdP)  
ON DELETE CASCADE  
ON UPDATE CASCADE,  
FOREIGN KEY (IdU) REFERENCES registrato(IdU)  
ON DELETE CASCADE  
ON UPDATE CASCADE);
```

Scelte sui vincoli di integrità della tabella:

-Entrambe le foreign key hanno cascade sia in update che delete perché sia nel caso che cessi di esistere il registrato che ha messo "mi piace" al post, sia che cessi di esistere il post che viene apprezzato da x utenti/e, voglio che la/le righe a cui fanno riferimento i rispettivi valori nella tabella figlia vengano eliminati (stesso discorso per l'update).

VIEWS:

POST_LIKE

```
CREATE VIEW post_like AS
SELECT
post.IdP AS Post,
post.TitoloP AS TitoloP,
post.TestoP AS Testo,
blog.IdB AS Blog,
blog.Titolo TitoloB,
blog.IdU AS IDProp,
blog.IdUcollab AS IDCollab,
(SELECT COUNT(*) FROM apprezza WHERE apprezza.IdP=post.IdP) AS nLike
FROM post
      JOIN blog ON blog.IdB=post.IdB
ORDER BY nLike DESC
```

Ragione di esistenza di questa view:

- **Questa view permette di visualizzare tutti i dati necessari (tra cui il calcolo del numero di like per ciascun post) alla visualizzazione della top 9 dei post con più like all'interno della pagina home.php; questo permette di effettuare meno query/estrazioni di dati dal database di quelle che sarebbero necessarie senza la view.**

ARGBLOGVIEW

```
CREATE VIEW argblogview AS
SELECT
argblog.IdT AS IdT
argblog.IdSt AS IdSt
argblog.IdB AS IdB
blog.Titolo AS Titolo
```

```

blog.IdU AS IdU
blog.IdSf AS IdSf
FROM (argblog
      JOIN blog ON(( argblog.IdB= blog.IdB)))

```

Ragione di esistenza di questa view:

- **Questa view permette di visualizzare determinati dati relativi ai blog (come gli id del collaboratore e del registrato e il titolo del blog) con x argomenti e x sottoargomenti. Senza questa view sarebbero necessarie un maggior numero di query/estrazioni di dati dal database per tale visualizzazione. Questa view è utilizzata per il calcolo dei blog risultanti dalla ricerca di blog tramite argomento, nella pagina BlogvisionView.php (funzione search_arg).**

BLOG_NICK

```

CREATE VIEW blog_nick
SELECT
    blog.IdB AS IdB,
    blog.Titolo AS Titolo,
    blog.IdU AS IdU,
    registrato.Nickn AS Nickn,
    blog.IdSf AS IdSf,
    blog.IdFo AS IdFo,
    blog.IdUcollab AS IdUcollab,
    blog.Ereditato AS Ereditato
FROM blog JOIN registrato ON registrato.IdU=blog.IdU

```

Ragione di esistenza di questa view:

- Questa view permette di visualizzare il nickname dell'utente proprietario (blog.IdU) dei blog, assieme agli altri dati dei blog. **Questa view è utilizzata per il calcolo dei blog risultanti dalla ricerca di blog tramite Nickname del proprietario, nella pagina BlogvisionView.php (funzione search_nick). Senza questa view sarebbe necessario fare una query per risalire agli IdU degli utenti con determinato Nickname che**

hanno matchato la ricerca e, a quel punto, ricavare tramite una seconda query i singoli blog che appartengono a questi utenti. Nel caso ci fossero più Nickname che corrispondono alla ricerca, sarebbe necessario fare una SELECT dei blog per ognuno degli utenti e fare infine un merge di tutti gli oggetti risultati da ogni SELECT (Si otterrà un oggetto per ogni utente che ha matchato la ricerca).

Trattandosi di un merge di oggetti e non di un merge di semplici array, che dovrebbe ricorsivamente fare un merge per ogni nuovo oggetto calcolato, la complessità di calcolo rischia di diventare sconsigliata rispetto alla possibilità di creare una view in cui abbiamo per ogni blog il nickname dell'utente proprietario: una view non occupa spazio nella memoria del database e viene eseguita solo quando "chiamata", creando una "vista" sui dati.

Difronte alla scelta di cosa fare, ho preferito fare una view.

Tabella delle operazioni

Operazioni disponibili per gli Utenti registrati

| | |
|----|---|
| 1 | fare login/logout |
| 2 | cancellare il proprio account utente |
| 3 | modificare i propri dati utente |
| 4 | modificare la propria password |
| 5 | ereditare blog altrui (nessun limite) |
| 6 | creare blog (massimo 5) |
| 7 | commentare (nessun limite) |
| 8 | mettere like (nessun limite) |
| 9 | togliere like (nessun limite) |
| 10 | seguire un blog (nessun limite) |
| 11 | non seguire più un blog che seguiva (nessun limite) |
| 12 | creare post (nessun limite) |
| 13 | modificare il blog (cambiare colori di font e sfondo e modificare il titolo del blog) (potere riservato a Autore e collaboratore) |
| 14 | cancellare i commenti (potere riservato a: Autore, collaboratore e proprietario del commento) |
| 15 | cancellare i post (potere riservato ad Autore e collaboratore) |
| 16 | cancellare un blog oppure farlo ereditare al collaboratore (potere riservato ad Autore) |

| | |
|----|--|
| 17 | Visualizzare i blog (post inclusi nel blog) |
| 18 | Nominare collaboratore(potere riservato ad autore) |
| 19 | Rimuovere collaboratore (potere riservato ad autore e collaboratore) |

Operazioni disponibili per gli utenti non registrati/visitatori

| | |
|----|---|
| 20 | effettuare ricerche |
| 21 | visualizzare l'homepage |
| 22 | fare Sign-up (isciversi) |
| 23 | visualizzare i blog ottenuti dai risultati di ricerca (nessuna azione oltre la visualizzazione è concessa) e quelli presenti nella top 9 della homepage |

Tabella business rules

| |
|--|
| Un blog può essere creato solo da un utente registrato |
| Non esistono registrati minorenni o con età che supera i 100 anni |
| Non possono esserci due nickname, numeri di telefono, documenti e email identici tra i registrati |
| Un utente registrato può creare al più 5 blog e non crearne affatto |
| Per ogni blog un utente registrato può creare infiniti post |
| Per ogni post ci possono essere infiniti commenti |
| Un commento è associato ad un post e può essere scritto da un utente registrato |
| Un post può avere associate al più 3 immagini o non averne affatto |
| Un utente che non ha fatto l'accesso può solo visualizzare (blog, post e homepage) e non può interagire finché non crea un account o non fa accesso. |
| Un blog può avere un collaboratore scelto dal proprietario del blog |
| Alla creazione del blog l'autore deve scegliere almeno un tema in cui classificarlo e può scegliere eventuali sotto argomenti |
| Un blog può avere solo un colore di sfondo e un colore di font attivi allo stesso momento. |

Guida alla cartella "thoughts"

Struttura della cartella:

- cartella **relazione e db estratto**: contiene la relazione del progetto e l'sql del database estratto
- cartella **immagini**: contiene le immagini utilizzate sul sito
- cartella **upload**: contiene le immagini che caricheranno gli utenti
- cartella **asset** si divide in 5 sottocartelle:
 - **bootstrap 4.5.3** : libreria bootstrap, non toccare
 - **bootstrap-select-1.13.14** : libreria bootstrap per il selectpicker, jquery plugin per bootstrap, non toccare
 - **popper**: contiene estrazione libreria popper, non toccare
 - **css**:contiene tutti i file css per le impostazioni di colori di sfondo e font e in più un file "**generic.css**" che contiene classi di stile generali per tutto il sito
 - **js**: contiene il file **my_script.js**, che contiene tutte le chiamate ajax e jquery del sito
- cartella **views**: contiene i body di tutte le pagine del sito, ovvero tutto ciò che verrà mostrato al cliente; contiene inoltre i file di header e footer.
- **File rimanenti** (cito i **fondamentali** per la funzione del sito):
 - *Page.php* "costruisce" tutte le pagine del sito tramite chiamate ai file di header, footer e body. Contiene inoltre la dichiarazione di variabile di connessione e la chiusura della connessione.
 - *include.php* contiene session_start() e l'inclusione del file di connessione al db "*connect.php*"
 - *connect.php* contiene la connessione al database.
 - *funzioni.php* contiene tutte le funzioni riutilizzate su più pagine (buona parte dei controlli degli input) e la funzione per controllare e inserire le immagini caricate nella cartella upload.
 - *index.php* contiene un indirizzamento alla pagina Home.php
 - *logout.php* permette l'operazione di logout.

I file **restanti** (esclusi quelli sopracitati) sono file php che conterranno la "costruzione" di tutte le pagine (funzione index che tramite page.php costruisce la pagina) e le funzioni (richiamate tramite evento ajax) necessarie per quella specifica pagina: saranno le **pagine principali** del sito.

Scelte di progetto

- Ho optato per l'utilizzo di un approccio PHP orientato agli oggetti (oop)
- Ho deciso di **inserire le password nel database già codificate dalla funzione php di hashing `password_hash`** (per approfondimenti leggere fine pagina 20-inizio 21 oppure leggere la documentazione inerente (nell'ultima sezione della relazione)).
- I **blog possono essere *dati in eredità***, ovvero i diritti possono essere ceduti agli utenti che per tale blog svolgono la funzione di collaboratore.
Questo atto è sempre svolto con la consapevolezza dell'utente proprietario del blog: questo viene **avvisato nei momenti in cui il blog "rischia" di essere dato in eredità**, ovvero nel momento in cui l'utente proprietario decide di **cancellare un blog** su cui esiste un collaboratore, oppure quando decide di **cancellare il proprio account** e sui suoi blog esistono dei collaboratori.
Il proprietario ha sempre il potere, prima di procedere in queste due operazioni, **di rimuovere il/i collaboratori dai/dal suo/suoi blog, o addirittura di inserire nuovi utenti collaboratori in modo che i/il suoi/suo blog non vengano del tutto persi.**
L'utente collaboratore, che a sua volta si trova improvvisamente ad essere proprietario di un blog su cui prima collaborava, ha il potere di cancellare il blog se non è interessato a gestirlo, di mantenerlo se è di suo interesse, o a sua volta cederlo in eredità ad altri.
- **Ho deciso di NON rendere modificabili i temi e sottotemi di un blog dopo la sua creazione:** il motivo è che avendo creato un meccanismo di **"fidelizzazione" ai blog** da parte dei singoli utenti tramite il "segui il blog" (che provoca un *collegamento* più rapido a questi tramite la comparsa di essi nella pagina personale Profile.php) **mi sembrava non corretto dare la possibilità agli utenti proprietari di blog di cambiare completamente le loro tematiche e sotto tematiche: se un utente decide di seguire un blog vuol dire che apprezza i tipi di contenuti che quel blog propone; dando la possibilità di modificare le tematiche dei blog, degli utenti potrebbero ritrovarsi dal giorno alla notte a seguire blog a cui non avrebbero mai messo un "segui" consapevolmente.**
Ad esempio, Pippo segue il blog di Rocco, che pubblica ricette di cucina che lui apprezza; **il giorno dopo Pippo ritrova tra i suoi "blog seguiti" un blog che nel migliore dei casi cambia**

completamente argomento, e nel peggiore pubblica contenuti fake, spam e ingannevoli senza che possa comprendere come un suo "seguì" sia potuto capitare su un tale blog.

Questo è uno dei difetti che, in vece di "cliente", ho riscontrato nel social network *Facebook*. Mettere like ad una pagina su Facebook equivale ad aumentare le possibilità che i post pubblicati su essi compaiano nella propria bacheca.

Una meccanica negativa creatasi su Facebook è la "vendita delle pagine": pagine con un alto numero di like ricevono offerte monetarie in cambio della cessione dei diritti sulla pagina in favore dei compratori.

In alcuni casi questa vendita va a buon fine e le pagine vengono riconvertite in pagine da contenuti spam, pubblicità, phishing, link malevoli e fakenews, che grazie all'elevata risonanza ottenuta dal numero di like (ottenuti dalla gestione precedente), si diffondono più efficacemente: un utente, ignaro di tali cambiamenti, può ritrovarsi nella bacheca questi tipi di contenuti provenienti da pagine a cui non avrebbe mai messo **consapevolmente** "mi piace".

Per un principio simile a quello precedentemente presentato dei "se offri un servizio sei responsabile della scelta di aver deciso di offrirlo, e se qualcuno ne usufruisce va mantenuto", quando un blog sceglie le tematiche da affrontare le deve mantenere, e mantenere coerenza verso sé stesso.

Se un utente preferisce parlare di altri argomenti può sempre creare un nuovo blog, mentre se un utente si trova a seguire blog che non avrebbe consapevolmente seguito questo può creare un danno di immagine al sito e farne perdere la credibilità.

Questo motivo mi ha portato a fare tale scelta.

- I like sui post possono essere messi da **qualsiasi** utente in **sessione**;
- Solo gli utenti in sessione che non sono proprietari e collaboratori di un blog X possono **seguire** quello specifico X blog.
- Ho deciso di non legare direttamente i post all'utente che li scrive perché nella mia **concezione i post sono elementi che caratterizzano il blog stesso a prescindere da chi in quel momento lo amministra.**

I collaboratori e proprietari dei blog possono creare post sui blog su cui in quel momento hanno potere, e i post da quel momento saranno elementi del blog.

Scelte per i controlli degli input

Controlli su input tramite funzioni (contenute nel file "funzioni.php"):

- `validate_nik_server($nickn_)` → controllo che la lunghezza del nickname sia compresa fra 4 e 11; se ciò è vero seleziono attraverso la query `checknick` tutti i nickname e controllo che quello fornito dall'utente non combaci con altri (nickname è unico per ogni utente)
- `validate_nick()` → funzione richiamata solo all'interno di `SignUp` in quanto è presente un bottone per verificare nell'immediato la disponibilità del nickname senza fare il submit.
- `Space($string)` → controlla tramite funzione regolare che non ci siano caratteri di spazio nei vari input.
- `checkemail($email)` → viene applicato all'input il filtro `FILTER_SANITIZE_EMAIL`, che rimuove automaticamente tutti i caratteri illegali da una mail, ma non ne verifica la correttezza del formato; per questo motivo viene in seguito applicato il filtro `FILTER_VALIDATE_EMAIL`. A questo punto, tramite la query `$checkmail` vengono estratte dal database tutte le email e se qualcuna di queste combacia con quella inserita in input, viene comunicato all'utente che quella mail è già in uso.
- `psw_check($psw_, $pswcheck_)` → è una funzione che controlla che la password fornita in input non sia più piccola di 6 o più grande di 11 caratteri, che contenga almeno una cifra, una lettera maiuscola e una lettera minuscola. A quel punto compara i due input forniti all'interno dei campi "inserisci password" e "reinserisci password" e se corrispondono il controllo va a buon fine.
- `validate_mobile($telefono)` → Questa funzione verifica, tramite funzione regolare, che il numero di telefono inserito abbia un formato valido per un numero di telefonia mobile italiano.

R.E → `/^(([+]|00)39)?((3[1-6][0-9]))(\d{7})$/` → Matcha una o nessuna occorrenza della sequenza 0039 oppure +39; poi matcha il 3 seguito da un numero compreso tra 1 e 6, seguito a sua volta da un numero compreso tra 0 e 9; a sua volta seguono 7 numeri;

L'uso di questa funzione regolare rende superfluo un controllo sul "tipo" di dato, che altrimenti sarebbe stato controllato tramite la funzione `php is_numeric()`.

Ho scelto di omettere la possibilità di inserire un numero di telefonia fissa per via di due fattori:

- in un contesto realistico un numero di telefono fisso va bene per quei tipi di servizi e-commerce che applicano tariffe speciali, promozioni etc. verso clienti attraverso un servizio di call center;
in un contesto di siti di blog, similamente a come fanno i maggiori social network, il numero di telefono fisso non ha scopi utili se non quello di immagazzinare dati dell'utente: ad esempio, per servizi di "avviso di notifica", "recupero password" e altri viene utilizzato il numero di telefonia mobile.

Realizzare una funzione regolare che prenda i primi 3 numeri (prefissi regionali di telefonia fissa) da una ipotetica lista realistica di tutti i prefissi regionali esistenti, oltre che essere energicamente dispendioso, non porterebbe alcun vantaggio per i motivi espliciti sopra. In termini di costi-benefici, i costi batterebbero i benefici. Si veda: <http://www.comuni-italiani.it/tel/>

- `valid_num_docu($documento)` → questa funzione verifica, tramite funzione regolare, che il numero identificativo del documento di riconoscimento inserito abbia un formato valido per i seguenti tipi di documenti italiani:
 - **patente**: 10 caratteri totali ("U1" + 7 caratteri alfanumerici + 1 carattere alfabetico) → `^\bU1[A-Z0-9]{7}[A-Z]\b$`
 - **carta d'identità**: 9 caratteri totali (2 lettere – 5 numeri – 2 lettere) → `^\b[A-Z]{2}\d{5}[A-Z]{2}\b$`
 - **Passaporto**: 9 caratteri totali (2 alfabetiche - 7 numeriche) → `^\b[A-Z]{2}\d{7}\b$`

In realtà in Italia esistono altri tipi di documenti d'identità (di cui specificherò **accanto il motivo per cui sono stati esclusi dalle possibilità d'input per l'utente**);

fonte: <http://www.carabinieri.it/cittadino/consigli/tematici/giorno-per-giorno/documenti/documenti#:~:text=%E2%80%9CSono%20equipollen ti%20alla%20carta%20d,di%20altra%20segnatura%20equivalente%2C%20rilasciate>

Estratto della fonte: "Il decreto del 2000 [1] stabilisce quanto segue:

«Sono equipollenti alla carta d'identità il passaporto, la patente di guida,

- la patente nautica, →formato non reperibile sul web
- il libretto di pensione→il numero si può conoscere solo tramite INPS= formato non reperibile sul web
- il patentino di abilitazione alla conduzione di impianti termici, →formato non reperibile sul web
- il porto d'armi, →formato non reperibile sul web
- le tessere di riconoscimento (purché munite di fotografia e di timbro o di altra segnatura equivalente, rilasciate da un'amministrazione dello Stato».)→arbitrario, non specificato= formato non reperibile sul web.

- `unique_docum($documento)` → funzione che verifica l'unicità del documento all'interno del sistema.

- `save_image($image)` → all'interno di suddetta funzione esiste un controllo per verificare che l'immagine passata in input tramite la funzione php `getimagesize()`: nella funzione restituisse false questo dimostra che non è un file immagine, e faccio restituire alla main function il valore null. Ciò provocherà ugualmente la pubblicazione del post ma senza immagini/file allegate.

Controlli su input senza funzioni:

- a tutti gli input **testuali** viene applicata la funzione php `stripslashes()` che rimuove eventuali backslash.
- Per tutti gli input **testuali** che non prevedono parole multiple (es: nome e cognome) applico la funzione `Space()` da me definita (pagina 27)
- Input "Nome" → tramite la funzione php `iconv_strlen()` controllo la lunghezza in caratteri: il campo "nome" nella tabella registrato prevede una grandezza massima di 30 caratteri, quindi tramite php controllo che l'input passato sia minore di 30 e maggiore di 3 (supponendo non esistano nomi così corti)
- Input "Cognome" → tramite la funzione php `iconv_strlen()` controllo la lunghezza in caratteri: il campo "cognome" nella tabella registrato prevede una grandezza massima di 20 caratteri, quindi tramite php controllo che l'input passato sia minore di 20 e maggiore di 3 (supponendo non esistano cognomi così corti)
- Input "Età" → tramite operatori maggiore e minore controllo che l'età passata in input sia quella di un maggiorenne (da 18(compresi) in poi) e **non** abbia l'improbabile età di 100 anni (da 100(compresi) i poi);

faccio un ulteriore controllo per constatare che il dato passatomi sia numerico con la funzione php is_numeric()).

Screenshot delle pagine

Home.php (visualizzazione dell'utente loggato)

Thoughts

Torna al ProfiloLogout

Benvenuto su Thoughts!

Scrivi tutto ciò che ti viene in mente o cerca opinioni interessanti!

Cerca

Turismo Sostenibile

Se ne parla sempre più spesso, la parola "turismo sostenibile" è sulla bocca di tanti. Ma cos'è davvero questa forma di turismo? E come si può praticare? Scoprendo insieme le abbiamo scritto tante volte, e sono sempre di più le persone che ne parlano: ma sappiamo dire con esattezza cos'è il turismo sostenibile? Fate su, e allora vi presentiamo la nostra guida pratica a questo modo di viaggiare. Partiamo dalla definizione, data dalla stessa Organizzazione Mondiale del Turismo: turismo capace di soddisfare le esigenze dei turisti di oggi e delle regioni ospitanti prevedendo e accrescendo le opportunità per il futuro. Tutte le risorse dovrebbero essere gestite in modo tale che le esigenze economiche, sociali ed estetiche possano essere soddisfatte mantenendo l'integrità culturale, i processi ecologici essenziali, la diversità biologica, i sistemi di vita dell'area in questione. I prodotti turistici sostenibili sono quelli che agiscono in armonia con l'ambiente, la comunità e le culture locali. In modo tale che essi siano i beneficiari e non le vittime dello sviluppo turistico.

2 mi piace

Post dal blog:

"Turismo -"

gestito da Rokky33

Rinvaso in corso!

Che fatica, ma il risultato è valso la pena!

1 mi piace

Post dal blog:

"Pianta Grasse: quali sono e come curarle"

gestito da nicky46

Vantaggi di consumare meno sale

Buone notizie per la salute pubblica. Tra il 2016 e 2019 quasi 5 italiani su 10 hanno ridotto la quantità di sale assunta a tavola. Lo rivelano i dati dell'Istituto Superiore di Sanità. Un risultato incoraggiante, viste le problematiche che un consumo eccessivo di sale comporta per il nostro benessere. Un corretto ed equilibrato uso del sale è di estrema importanza. Infatti, se da una parte il sale è fondamentale per l'organismo, dall'altra un suo consumo eccessivo è una delle cause più comuni delle malattie cardiovascolari. Il sale apporta sodio e cloro, due elementi fondamentali nella regolazione dell'equilibrio acido-base dell'organismo nel bilancio idrico (vale a dire la distribuzione dei liquidi nonché il volume di sangue nell'organismo). Il sodio, in particolare, ha funzioni molto importanti regolando la quantità di acqua presente nel sangue e tra le cellule dei tessuti. Inoltre, influenza la contrazione muscolare e la trasmissione dell'impulso nervoso.

1 mi piace

Post dal blog:

"Una mela al giorno"

gestito da Rokky33

Regio Torino ALive, concerto

Protagonista l'Orchestra d'archi Teatro Regio Torino, diretta da Andrea Baum, che esegue la Sinfonia per archi n. 10 di Felix Mendelssohn-Bartholdy. Intermezzo da Cavalleria rusticana di Pietro Mascagni, Soggetti op. 30 di Edward Elgar e l'Adagietto dalla Quinta Sinfonia di Gustav Mahler. Il concerto rientra nell'iniziativa digitalisimmentamento, lanciata dall'AnfoIs e alla quale hanno aderito le Fondazioni lirico-sinfoniche italiane, che prevede una ricca stagione in streaming dai siti web e dalle pagine Facebook dei Teatri, nonché sulla web tv dell'AnfoIs (anfois.it/webtv) e sul sito dell'ANISA (anisa.it).

1 mi piace

Post dal blog:

"Cultura e novità dal mondo"

gestito da nicky46

Quando si ingrassa di più?

Quando ingrossiamo di più: lo studio il nostro peso oscilla nel tempo e molti fattori possono essere all'origine di queste variazioni: lo stress l'età una malattia i cambiamenti delle abitudini. Tre pariali della vita molto specifici possono avere un impatto sulla nostra linea. Lo rivelano due meta-analisi condotte da ricercatori dell'Università di Cardiff (Regno Unito) e pubblicate su Obesity Reviews. Il lavoro di ricerca si è concentrato su: impatto della gentrificazione; conseguenze dell'accesso all'istruzione superiore primo lavoro.

1 mi piace

Post dal blog:

"Una mela al giorno"

gestito da Rokky33

Di cosa parliamo?

Cosa dobbiamo intendere per inquinamento? Negli ultimi anni l'uso di questa parola si è molto esteso, tanto che nel linguaggio comune la parola inquinamento è spesso usata come sinonimo di ambiente sporco. L'inquinamento vero e proprio consiste invece nell'introduzione diretta o indiretta in un ambiente di sostanze o onde di energia capaci di trasformare gli equilibri naturali producendo anche effetti sulla salute umana. Alcune di queste trasformazioni sono irreversibili nel medio o nel lungo periodo. L'inquinamento può essere provocato da fenomeni naturali – per esempio eruzioni vulcaniche, incendi, radioattività di alcune rocce – o da attività dell'uomo. In entrambi i casi, vengono immesse in un ambiente sostanze estranee e più o sostanze comuni ma in quantità tali che superano la capacità di digestione (degradazione e decomposizione) e assorbimento da parte di quell'ambiente: è il caso dell'eutrofizzazione negli ambienti acquatici o dell'eccesso di produzione di anidride carbonica che provoca l'effetto serra. Nell'ultimo secolo l'inquinamento provocato dalle attività umane ha di gran lunga superato l'inquinamento di origine naturale.

1 mi piace

Post dal blog:

"Abbiamo solo un pianeta"

gestito da Rokky33

New entry!

Non è magnifica?

1 mi piace

Post dal blog:

"Pianta Grasse: quali sono e come curarle"

gestito da nicky46

I miei progetti!

0 mi piace

Post dal blog:

"Modellare negli anni 2000"

gestito da nicky46

Vacanze all'estero per

Qui puoi trovare informazioni di viaggio pratiche sui visti e sui requisiti di accesso, sui regolamenti doganali e di quarantena e sui viaggi accessibili in Australia. Ci sono anche informazioni per aiutarti a pianificare il tuo viaggio, tra cui come arrivare e spostarsi in Australia, i tour che puoi fare mentre sei qui e le strutture in cui soggiornare. Offriamo inoltre alcuni consigli utili sui viaggi nelle zone più remote, suggerimenti, informazioni sui pericoli a cui prestare attenzione e chi contattare in caso di emergenza durante il tuo soggiorno. Scopri le nostre città, gli stati e i territori, il clima, i fusi orari, la storia, la fauna e la flora uniche al mondo e le informazioni sul visto australiano e i requisiti per l'ingresso nel paese.

0 mi piace

Post dal blog:

"Turismo -"

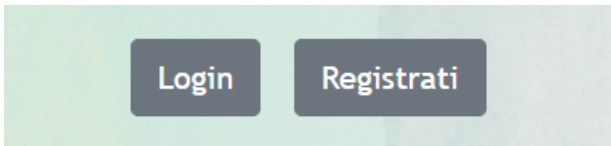
gestito da Rokky33

Torna in cima alla pagina

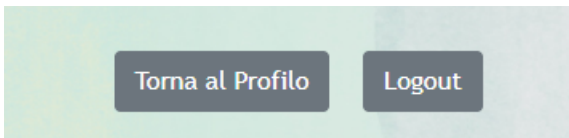
Sito realizzato da Simona Sette
matricola 544798
CDS Informatica Unimathica

L'utente non loggato vede la stessa identica pagina tranne per i bottoni della "barra di ricerca":

Bottoni mostrati all'utente visitatore:



Bottoni mostrati all'utente loggato:



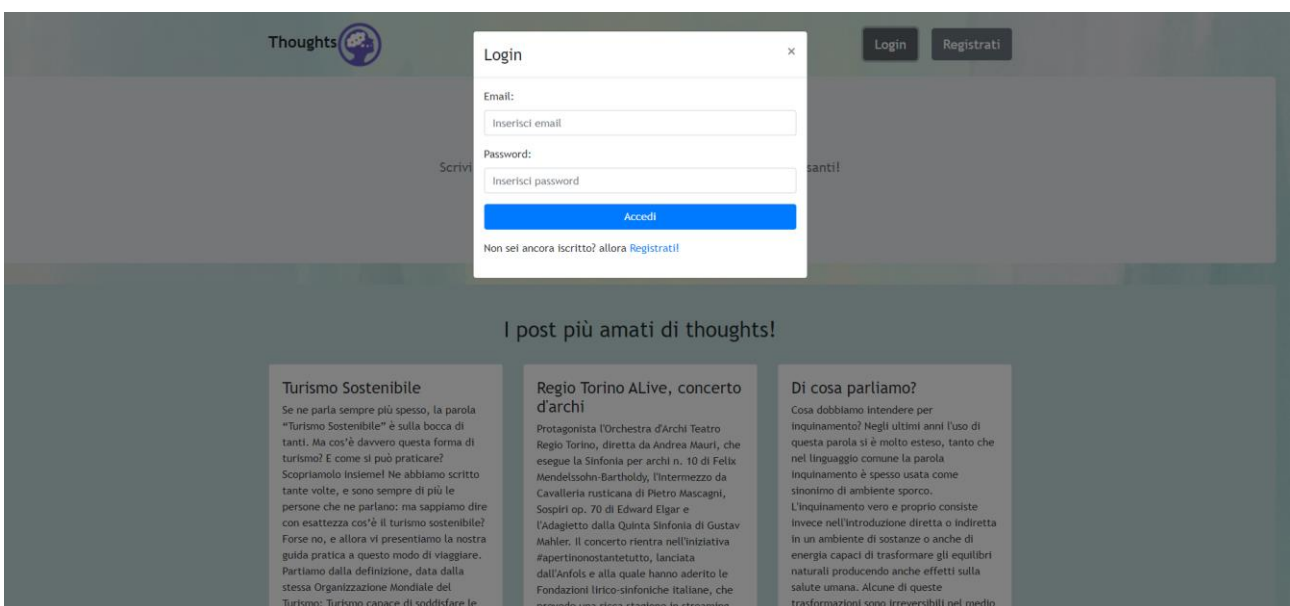
Sulla pagina è presente una "top 9" dei post con più like del sito, con annessi:

- Numero di like del post
- Titolo e testo del post
- i nickname dell'utente proprietario e collaboratore del blog su cui è postato il blog
- titolo del blog/link che porta al blog

Al centro della pagina c'è il bottone "cerca" che porta alla pagina cerca.php, dove possono essere effettuate ricerche di blog.

Infondo alla pagina è presente un simil bottone che riporta in cima (considerando che i post possono essere molto lunghi).

Login



È realizzato attraverso un modal bootstrap sulla pagina Home.php

Se un utente non dispone di un account trova qui un reminder della necessità di registrarsi per fare login, con annesso link (Registrati!) che porta alla pagina SignUp.php

In alternativa, per l'utente non registrato (e di conseguenza loggato) esiste il bottone "Registarti" sulla pagina che porta alla pagina SignUp.php

SignUp.php

Questa pagina è inaccessibile a un utente loggato (viene rimandato alla pagina Home).

Nella barra di navigazione è disponibile un bottone che porta alla home.

Vi sono le indicazioni (tramite paragrafi) per inserire i tipi di input adatti al singolo campo, quando necessari.

Il bottone "controlla disponibilità" permette di controllare la disponibilità del nickname proposto dall'utente (essendo univoci per ogni registrato) in modo che non debba ogni volta fare un submit di registrazione e avere messaggi di errore prima di poter riuscire a registrarsi.

Nell'eventualità che l'utente non controlli tramite bottone la disponibilità del nickname proposto, i controlli vengono comunque effettuati.

Thoughts [Home](#)

NickName:
Inserisci il tuo nickname e verificane la disponibilit *
controlla disponibilit 
*Se non ti va di verificare non ti preoccupare, lo faremo noi al posto tuo!

Nome:
Inserisci il tuo nome

Cognome:
Inserisci il tuo cognome

Et :
Inserisci la tua et 

Email:
Inserisci la tua email

Telefono:
Inserisci il tuo numero di telefonia mobile
*Sono accettati solo numeri di telefonia mobile.

Numero di Documento d'Identit :
Inserisci il tuo numero di documento
*Sono accettate: carta d'identit , passaporto o patente.

Password:
Inserisci la tua password
*La password deve contenere un carattere minuscolo, una cifra ed essere lunga minimo 6 caratteri e massimo 11 caratteri.

Reinserisci la password:
Reinserisci la tua password

Sesso:
Seleziona ▾

Accedi

Sito realizzato da Simona Sette
matricola 544298
CDS Informatica Umanistica

Profile.php

Pagina disponibile e visualizzabile **solo** per l'utente loggato;

Se un utente non è in sessione e prova ad accedere a questa pagina, viene reindirizzato alla home.

Thoughts

HomeCercaLogout

Area Personale di Rocco Giuseppe Ferdinando Lauria Iacovone

I tuoi blog

Una mela al giorno

Ricette di Rokky

Corpore sano in mens sana

Abbiamo solo un pianeta

Blog ereditati da altri utenti

Turismo

Crea un nuovo blog!

Blog a cui collabori

Non sei ancora collaboratore di nessun blog.

Blog che segui

Cultura e novità dal mondo

I tuoi dati

Nickname:
Rokky33

Nome:
Rocco Giuseppe Ferdinando

Cognome:
Lauria Iacovone

E-mail:
roccolauria@gmail.com

Cellulare:
3332565765

Documento:
CA92763CG

Età:
33

Sesso:
Maschio

Modifica i tuoi dati

Modifica la tua password

Il tuo account

*se decidi di cancellare il tuo account tutti i tuoi blog che hanno un collaboratore verranno ereditati da questo.
Se desideri cancellarli definitivamente rimuovi i collaboratori dai tuoi blog e poi prosegui.

Elimina il tuo account

Sito realizzato da Simona Sette
matricola 544298
CDS Informatica Umanistica

Questa è la pagina a cui un utente che entra in sessione (appena registrato/appena loggato) viene reindirizzato.

La barra di navigazione offre la possibilità di spostarsi sulla home, sulla pagina di ricerca blog e il bottone di logout.

Su questa pagina l'utente in sessione ha una completa visione dei dati di suo interesse: nella prima parte vengono mostrati i blog **creati** da lui e il bottone per crearne.

Il bottone scompare dopo aver creato il 5 blog (un utente può creare al massimo 5 blog). In questa sezione vengono anche mostrati i blog di sua proprietà perché lasciati in eredità a altri utenti. La sezione "blog ereditati da altri utenti non viene mostrata se l'utente non ha ricevuto nessun blog in eredità (oppure se li ha cancellati).

Esempio del medesimo utente post cancellazione blog ereditato:

I tuoi blog

[Una mela al giorno](#)

[Ricette di Rokky](#)

[Corpore sano in mens sana](#)

[Abbiamo solo un pianeta](#)

[Crea un nuovo blog!](#)

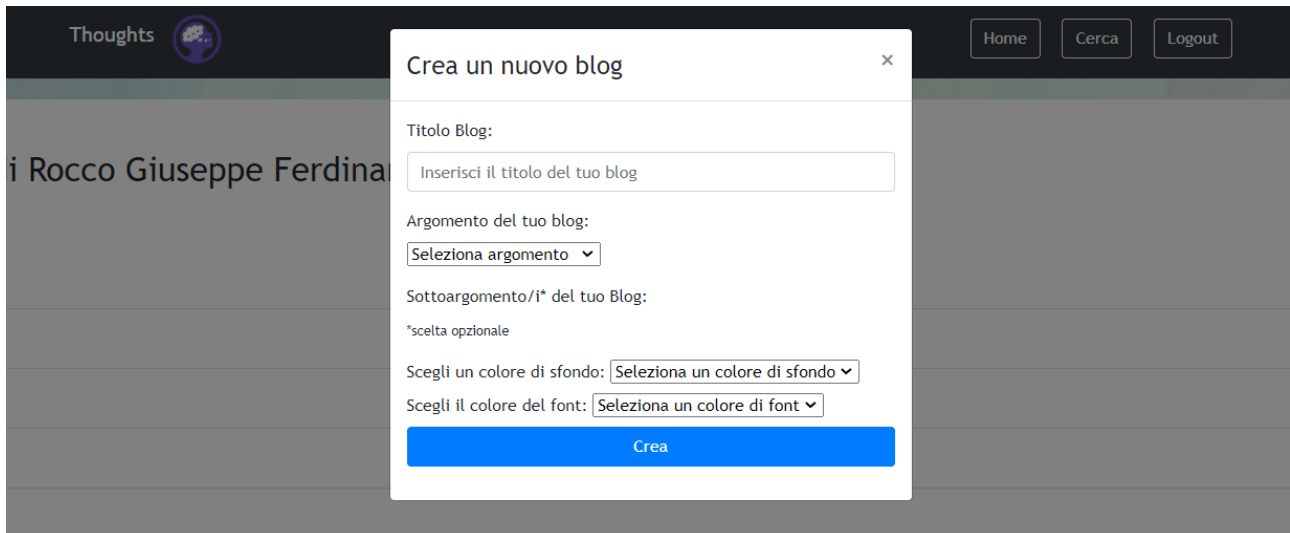
Segue la sezione "blog a cui collabori" e "blog che segui", dove compaiono i blog su cui l'utente in sessione è collaboratore e i blog che l'utente in sessione segue perché apprezza i contenuti di suddetto blog.

Tutta la sezione inerente ai blog offre la possibilità di spostarsi su tali blog tramite titolo/link, che porta alla pagina ChosenBlog.php relativa al blog selezionato.

Segue la sezione che riguarda i dati personali, dove vengono stampati i valori correnti dell'utente. L'utente può scegliere di modificare i propri dati (bottone relativo) tranne la mail, oppure modificare la password tramite relativo bottone.

Segue la sezione "il tuo account" dove è possibile cancellare il proprio account (tramite bottone) e, tramite paragrafo, l'utente viene avvisato delle conseguenze di tale azione.

“Crea un nuovo blog”

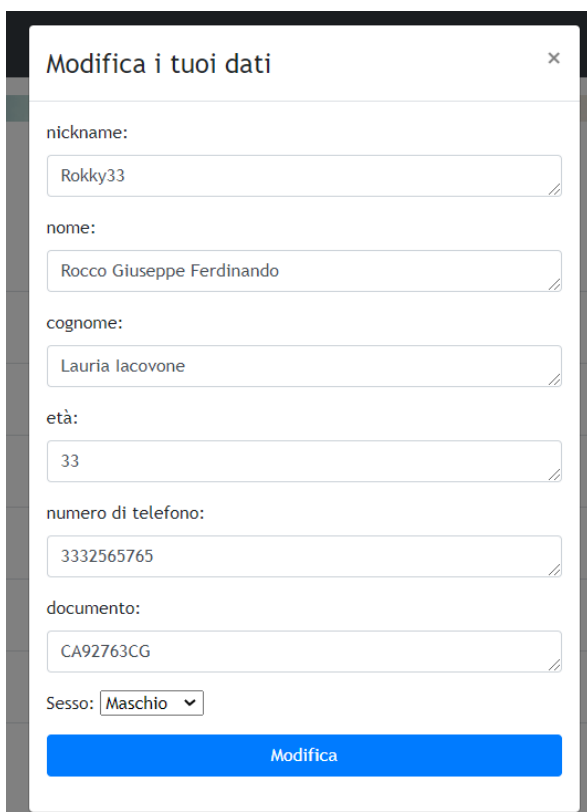


The screenshot shows a web application interface with a dark header. On the left, the word "Thoughts" is next to a brain icon. On the right, there are buttons for "Home", "Cerca", and "Logout". A modal window titled "Crea un nuovo blog" is open in the center. It contains the following fields: a text input for "Titolo Blog:" with placeholder "Inserisci il titolo del tuo blog"; a dropdown menu for "Argomento del tuo blog:" with "Seleziona argomento" selected; a text input for "Sottoargomento/i* del tuo Blog:" with a note "*scelta opzionale"; a dropdown menu for "Scegli un colore di sfondo:" with "Seleziona un colore di sfondo" selected; and a dropdown menu for "Scegli il colore del font:" with "Seleziona un colore di font" selected. At the bottom of the modal is a blue button labeled "Crea".

Quando l'utente spinge il bottone per creare un nuovo blog, gli si apre il model inerente a questa funzionalità; qui può digitare il titolo, decidere i colori di sfondo e font e decidere l'argomento del blog ed eventualmente scegliere dei sottoargomenti per esso tra quelli disponibili per tale argomento.

Una volta creato il blog, si viene reindirizzati al blog appena generato.

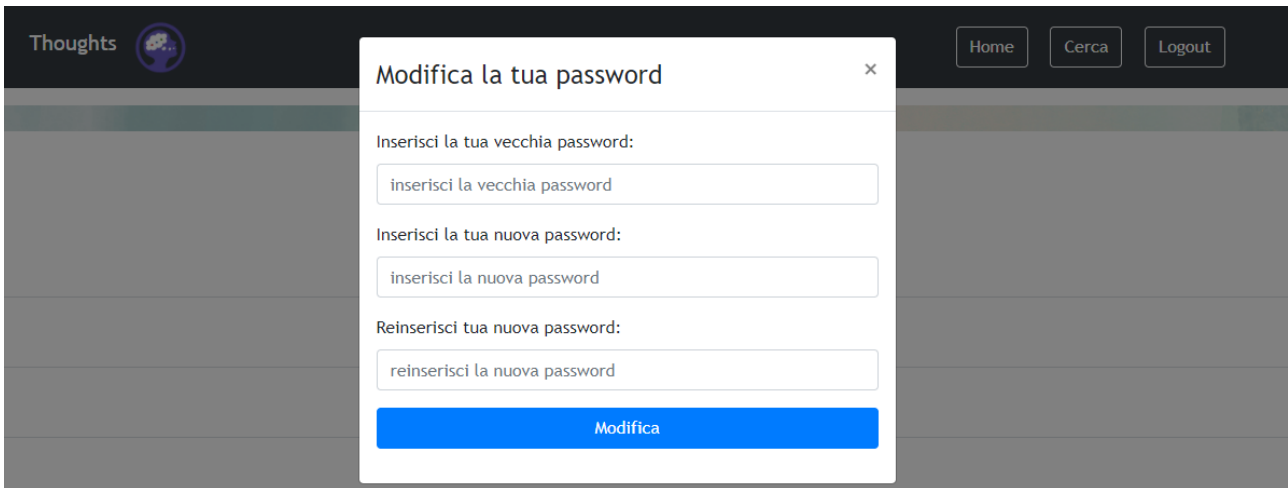
“Modifica i tuoi dati”



The screenshot shows a modal window titled "Modifica i tuoi dati" with a close button (X) in the top right corner. The form contains the following fields: "nickname:" with input "Rokky33"; "nome:" with input "Rocco Giuseppe Ferdinando"; "cognome:" with input "Lauria Iacovone"; "età:" with input "33"; "numero di telefono:" with input "3332565765"; "documento:" with input "CA92763CG"; and "Sesso:" with a dropdown menu showing "Maschio". At the bottom is a blue button labeled "Modifica".

Quando l'utente spinge il bottone per modificare i propri dati, gli si apre il model inerente a questa funzionalità. Gli vengono mostrati i dati correnti e lui può decidere quale modificare e se modificarli. Se viene effettuata una modifica (e supera i controlli) l'utente viene avvisato della buona riuscita dell'operazione e viene ricaricata la pagina in modo che le modifiche siano visibili nella sezione "I tuoi dati".

"Modifica la password"



The screenshot displays a web application interface. At the top, a dark header bar contains the word "Thoughts" next to a brain icon, and three buttons: "Home", "Cerca", and "Logout". In the center, a white modal box titled "Modifica la tua password" is open. It contains three input fields with placeholder text: "inserisci la vecchia password", "inserisci la nuova password", and "reinserisci la nuova password". A blue button labeled "Modifica" is positioned at the bottom of the modal.

Quando l'utente spinge il bottone per modificare la propria password, gli si apre il model inerente a questa funzionalità.

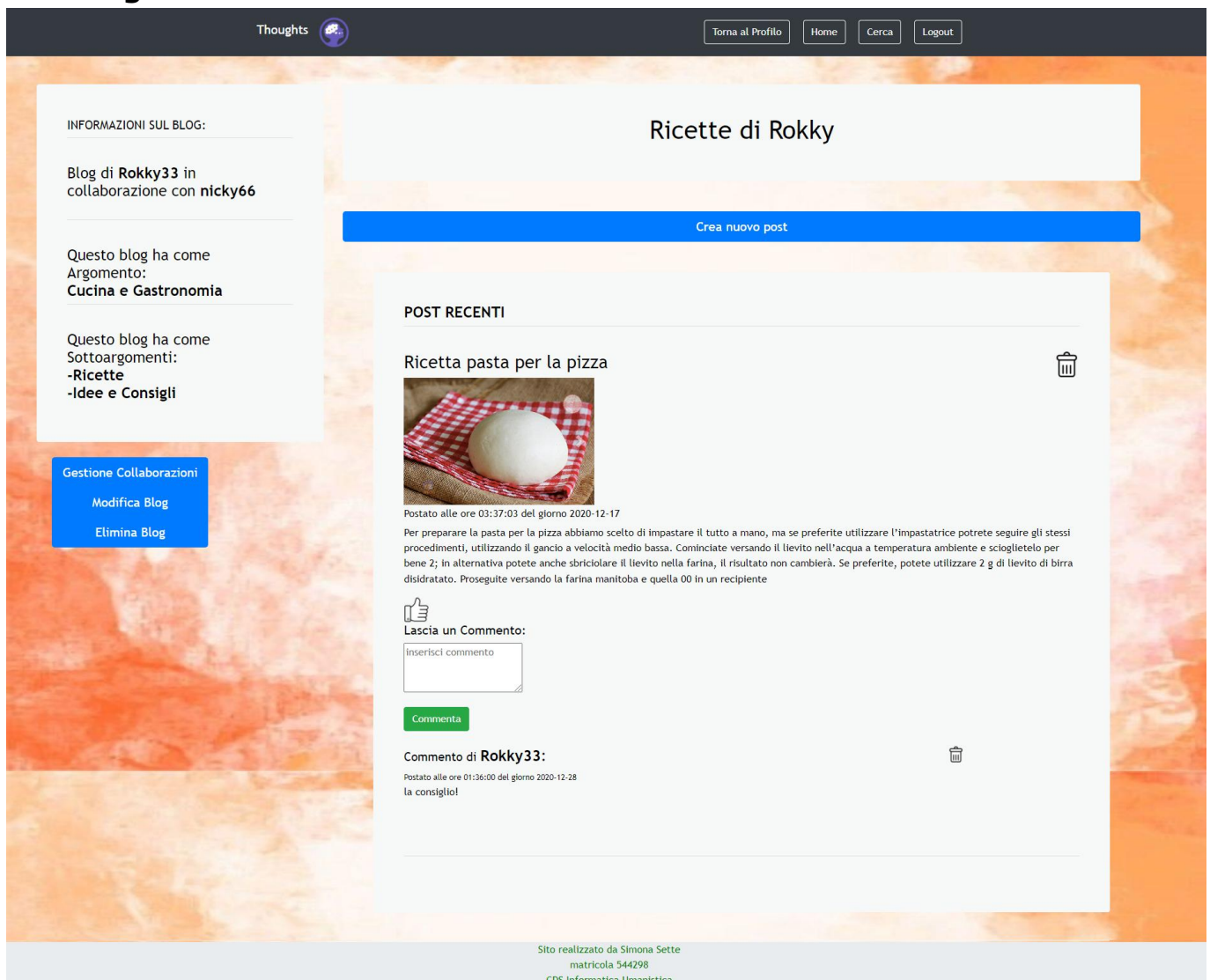
L'utente dovrà reinserire la password corrente, inserire una nuova proposta di password e reinserirla. Se la modifica supera i controlli l'utente viene avvisato della riuscita dell'operazione e la pagina viene ricaricata (altrimenti con un semplice `.modal(hide)` i dati inseriti nel modal resterebbero lì nel caso l'utente decidesse di riaprire il modal subito dopo).

"Elimina il tuo account" provoca la cancellazione istantanea dell'account (e dati correlati a quell'utente).

ChosenBlog.php

Questa pagina è accessibile sia ad utenti in sessione che utenti visitatori. Per questo motivo vengono mostrati bottoni diversi nella barra di navigazione in base al tipo di utente che la visita.

- Visione del blog da parte di un **utente in sessione e proprietario del blog**:



Differenze tra utente proprietario e utente collaboratore del blog:

L'utente collaboratore non ha a disposizione il bottone "elimina blog", in quanto ho deciso di non concedere questo diritto al collaboratore.

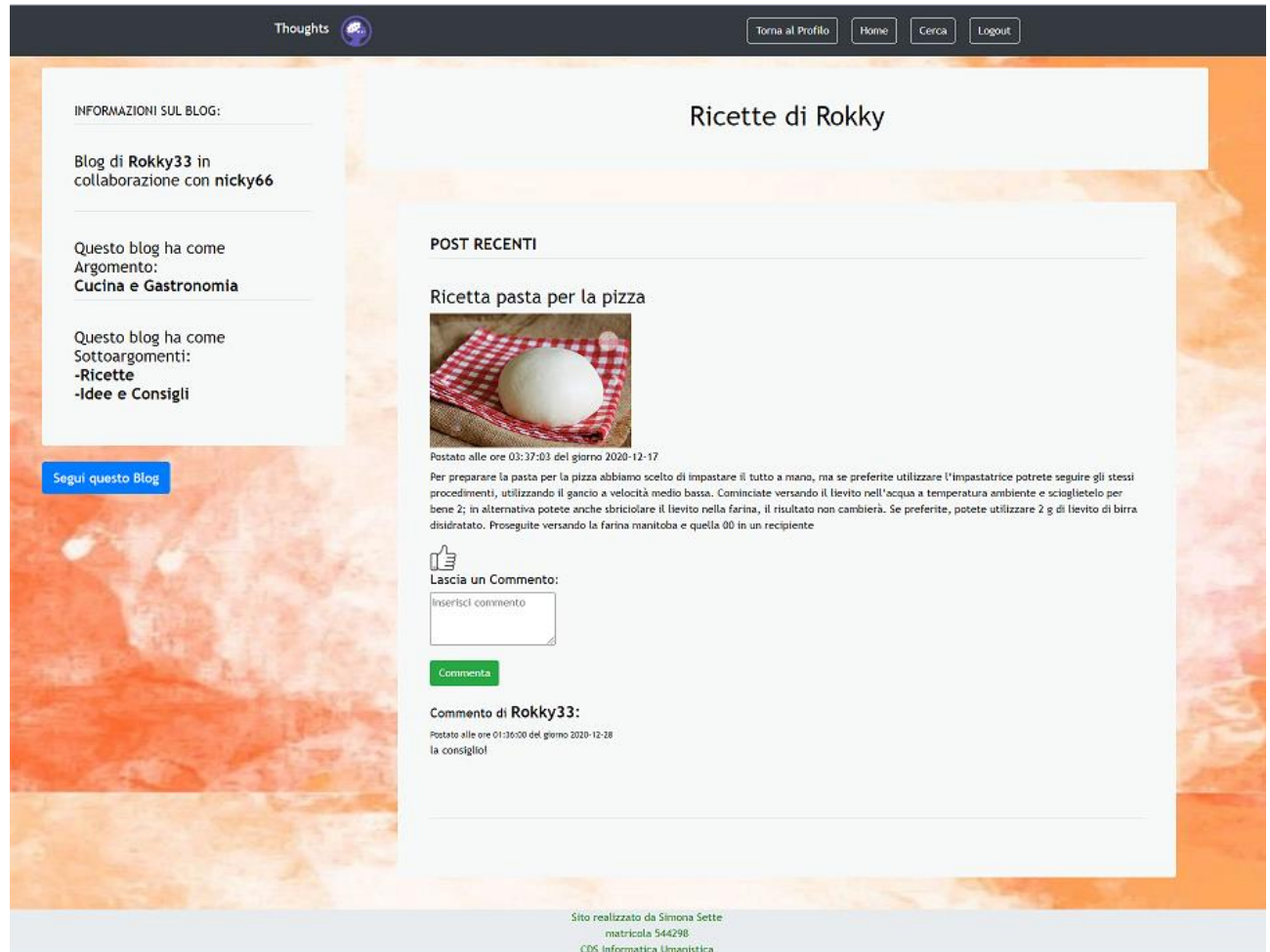
Es: quali bottoni di quelli sulla sinistra vede il suo collaboratore nicky66:



Per il resto, il collaboratore ha gli stessi diritti e poteri dell'utente proprietario.

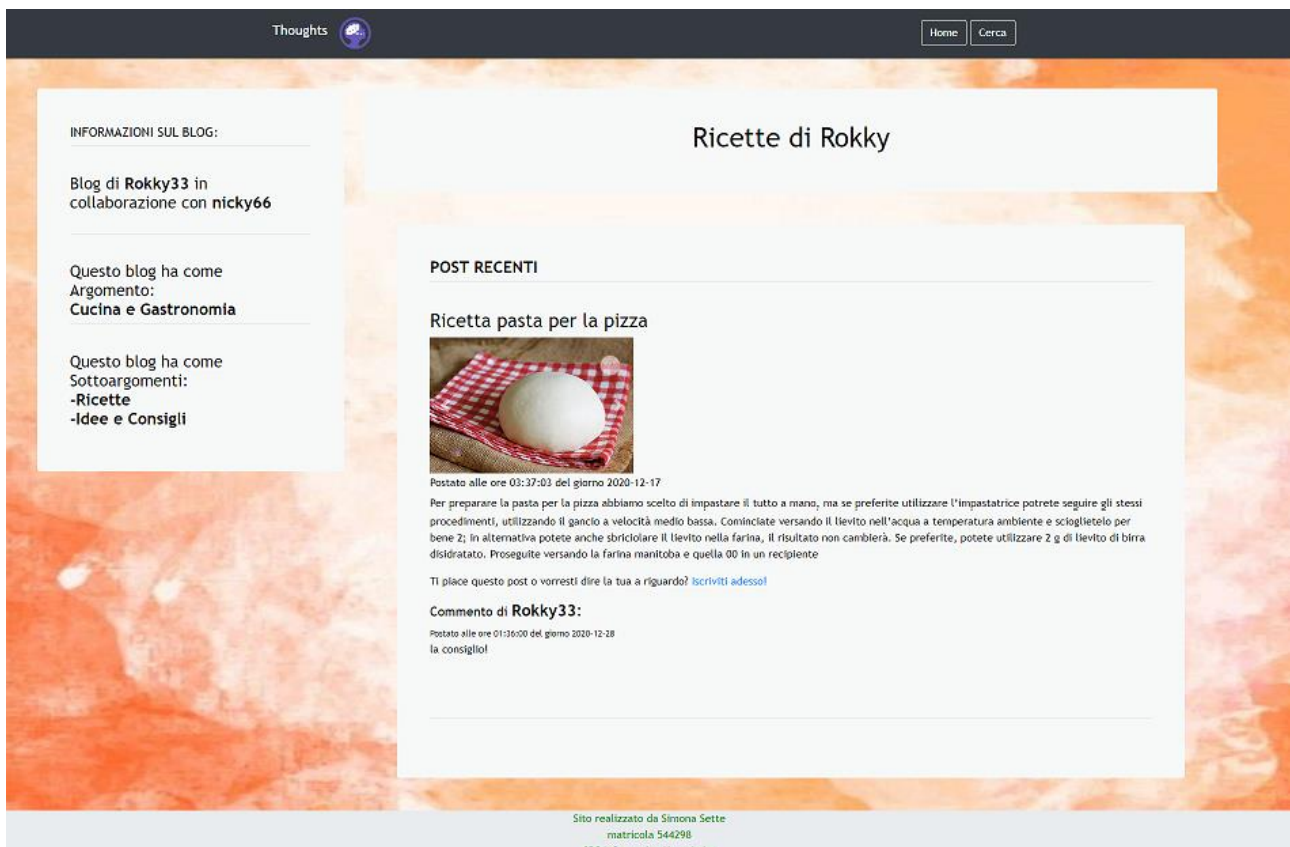
Sia il collaboratore che il proprietario possono mettere like ai post sui blog che gestiscono/possiedono **ma non possono seguire** blog gestiti da loro.

- Visione del blog da parte di un **utente in sessione diverso dal proprietario e collaboratore:**



Per ovvi motivi i bottoni legati alla gestione del blog non sono disponibili per l'utente loggato generico, ma può mettere like sui post, commentare, e seguire i blog (operazione disponibile solo per questo tipo di utente): il segui provoca la comparsa di tale blog (sottoforma di titolo/link) nella pagina privata dell'utente che comincia a seguire, in modo che tale utente abbia un collegamento più efficiente e diretto con i contenuti che evidentemente apprezza.

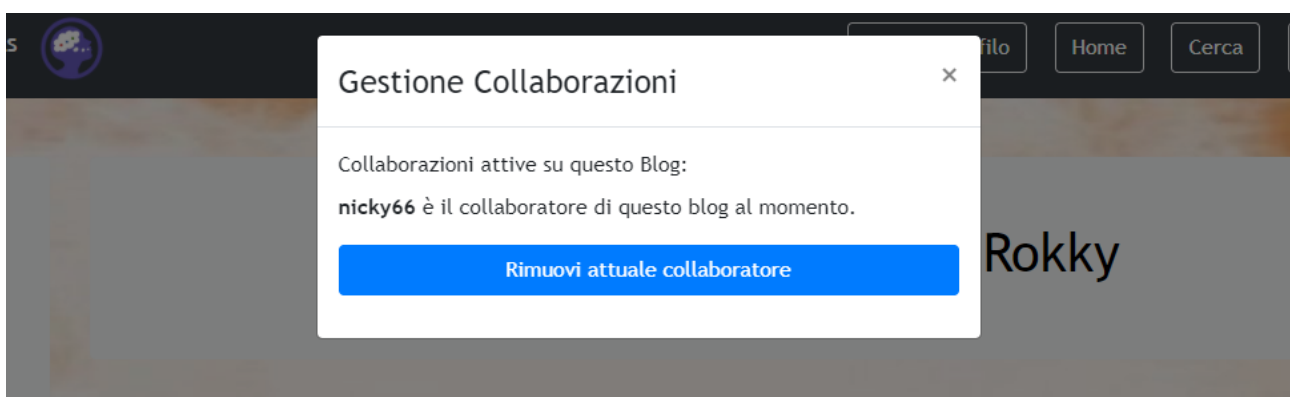
- Visione del blog da parte di un **utente visitatore (non in sessione)**:



Questo tipo di utente non ha poteri di interazione con i blog o i post: può soltanto visualizzare post, commenti e informazioni sul blog.

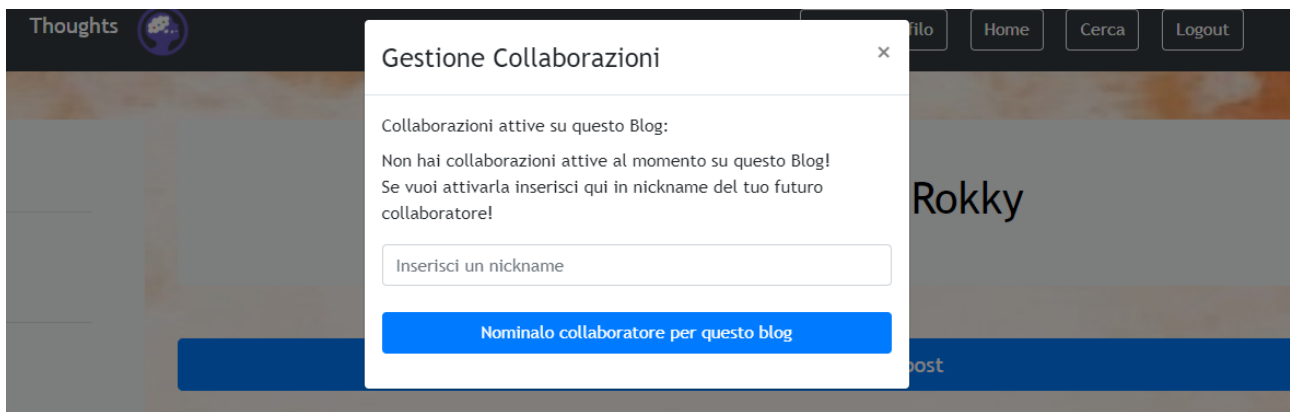
Viene inoltre sollecitato a interagire e gli viene fornito un link (iscriviti adesso) che porti alla pagina di registrazione al sito (SignUp.php).

“Gestione collaborazioni”



Se è presente un collaboratore per il blog, viene data la possibilità di rimuoverlo.

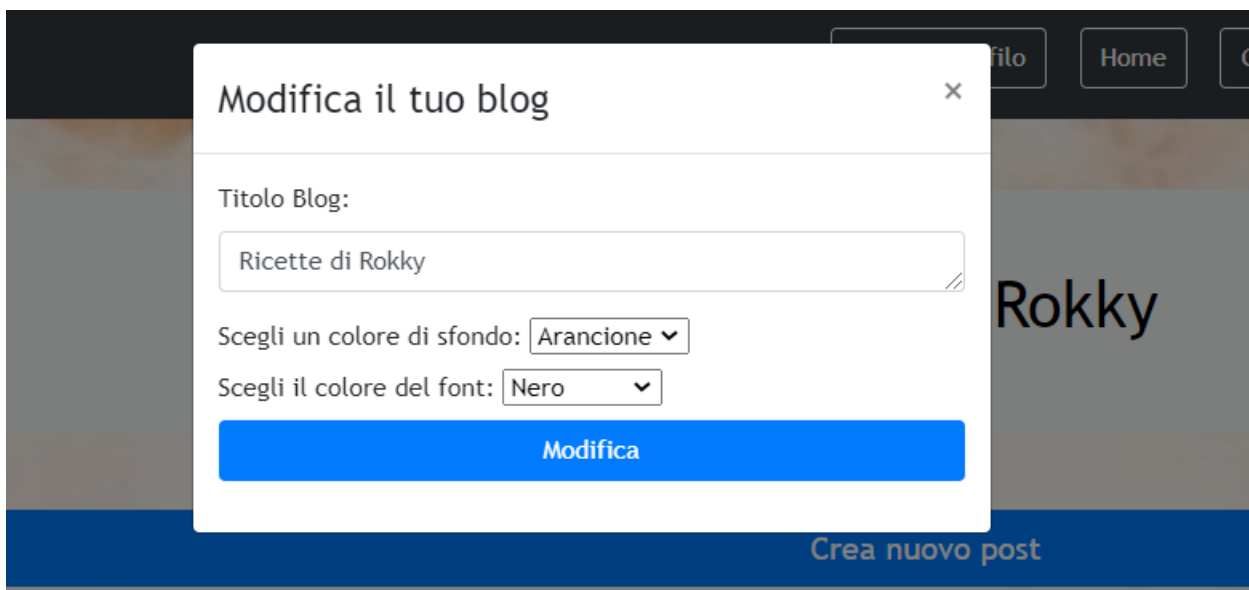
Se rimosso, la pagina viene ricaricata e nelle informazioni del blog in alto a sinistra si può notare come la riga inerente al collaboratore scompaia.



Se non è presente un collaboratore sul blog, viene data la possibilità di inserirlo (tramite inserimento nickname).

Se l'input supera i controlli, la pagina viene ricaricata e nelle informazioni del blog in alto a sinistra si può leggere la riga inerente al collaboratore.

“Modifica il tuo blog”

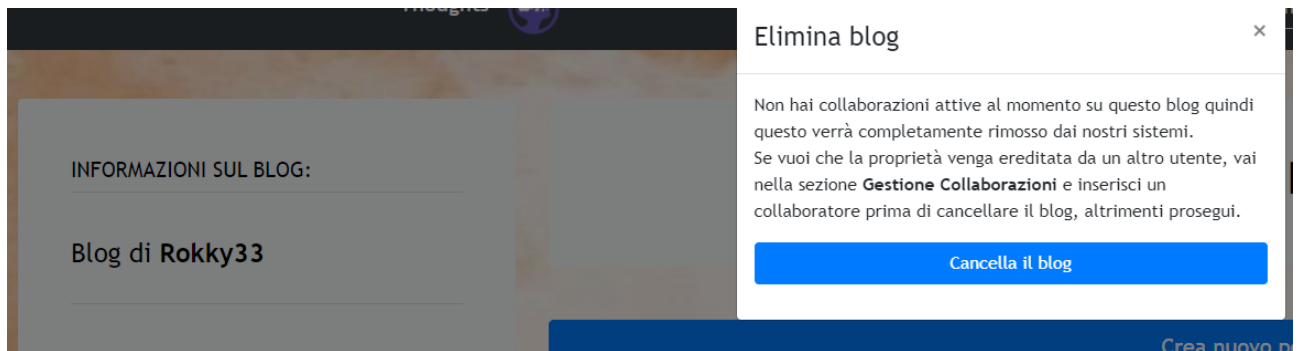


Qui è possibile modificare i colori di sfondo e font, oltre che il titolo del blog.

Se la modifica supera i controlli, la pagina viene ricaricata e sono visibili le modifiche.

“Elimina blog”

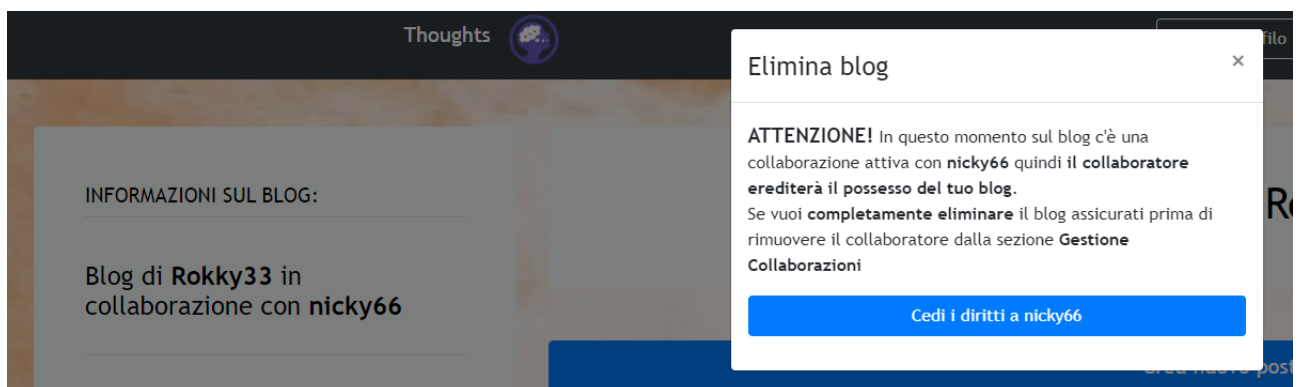
- Senza collaboratore:



Nel caso il blog NON abbia un collaboratore gli viene comunicato che il blog verrà cancellato definitivamente dal sito, a meno che prima di procedere non nomini un collaboratore (lasciandolo quindi in eredità ad esso).

Se procede, il blog viene rimosso definitivamente dai sistemi.

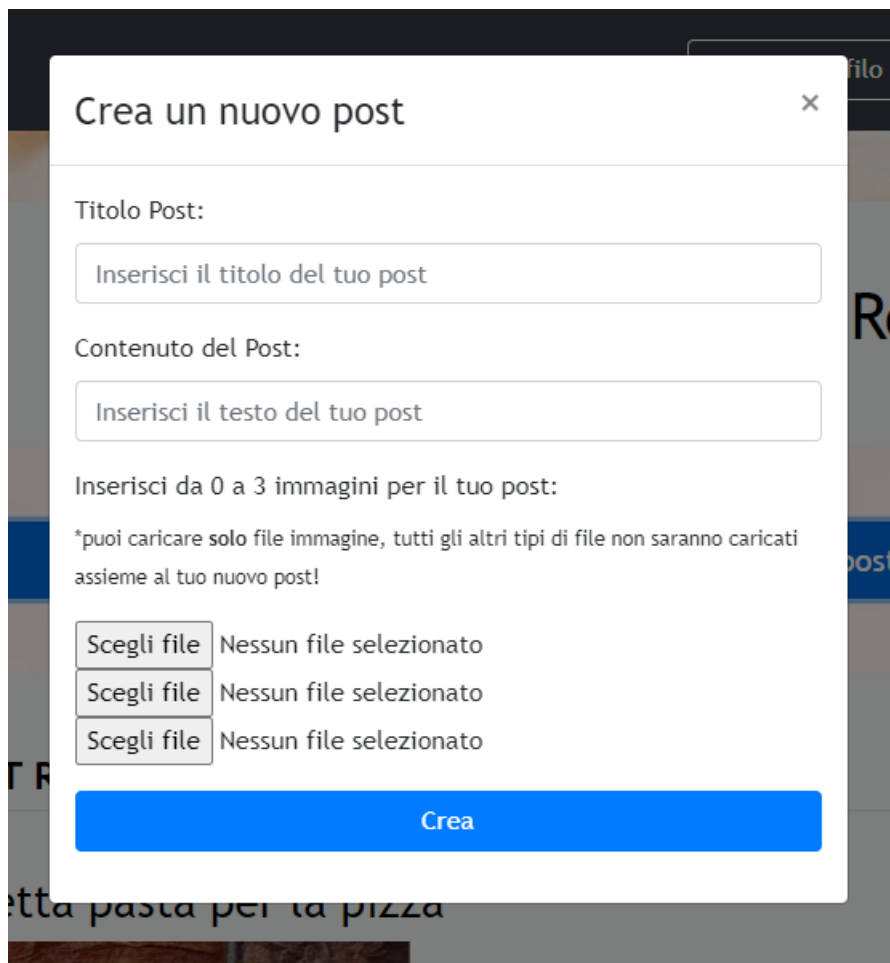
- Con collaboratore:



Nel caso il blog abbia un collaboratore gli viene comunicato che il blog verrà lasciato in eredità ad esso, a meno che prima di procedere non rimuova il collaboratore.

Se procede, il blog viene lasciato in eredità al collaboratore e il proprietario perde ogni diritto sul blog.

“Crea Blog”

A screenshot of a web application showing a modal window titled "Crea un nuovo post" with a close button (X) in the top right corner. The form contains three input fields: "Titolo Post:" with placeholder text "Inserisci il titolo del tuo post", "Contenuto del Post:" with placeholder text "Inserisci il testo del tuo post", and a section for images with the instruction "Inserisci da 0 a 3 immagini per il tuo post:" and a note "*puoi caricare solo file immagine, tutti gli altri tipi di file non saranno caricati assieme al tuo nuovo post!". Below this are three "Scegli file" buttons, each followed by the text "Nessun file selezionato". At the bottom is a large blue button labeled "Crea".

Crea un nuovo post

Titolo Post:

Inserisci il titolo del tuo post

Contenuto del Post:

Inserisci il testo del tuo post

Inserisci da 0 a 3 immagini per il tuo post:

*puoi caricare solo file immagine, tutti gli altri tipi di file non saranno caricati assieme al tuo nuovo post!

Scegli file Nessun file selezionato

Scegli file Nessun file selezionato

Scegli file Nessun file selezionato

Crea

Tramite questo model è possibile creare un nuovo post, con un titolo, testo e da 0 a massimo 3 immagini (quindi opzionali). L'utente viene inoltre avvisato del fatto che se provasse a inserire file non di tipo immagine, il file non sarà caricato assieme al post, ma il post sarà caricato comunque.

Una volta fatto, la pagina si ricarica mostrando il post appena creato in cima a tutti gli altri.



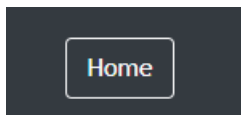
Su un blog è possibile cancellare post (potere del proprietario e collaboratore) e cancellare commenti su post (potere di: proprietario e collaboratore del blog su cui risiede il post, e proprietario del commento)

Cerca.php

Visualizzazione dell'utente in sessione:

The screenshot displays a web application interface with a dark header bar. On the left of the header is the text 'Thoughts' next to a small circular profile icon. On the right are three buttons: 'Torna al Profilo', 'Home', and 'Logout'. The main content area has a light blue background and contains three white search boxes. The first box is titled 'Ricerca i blog in base ad argomenti!' and contains a dropdown menu with 'Medicina e Salute' selected and a blue button labeled 'Cerca tramite argomento'. The second box is titled 'Ricerca i blog in base ad una parola!' and contains a text input field with the placeholder 'Inserisci una parola' and a blue button labeled 'Cerca tramite parola'. The third box is titled 'Ricerca i blog in base all'autore!' and contains a text input field with the placeholder 'Inserisci un nickname' and a blue button labeled 'Cerca tramite autore'. At the bottom of the page, there is a small footer with the text: 'Sito realizzato da Simona Sette', 'matricola 544298', and 'CDS Informatica Umanistica'.

Questa pagina è accessibile a ogni tipo di utente: l'utente visitatore visualizza la stessa pagina, cambiano solo i bottoni presenti nella barra di navigazione:



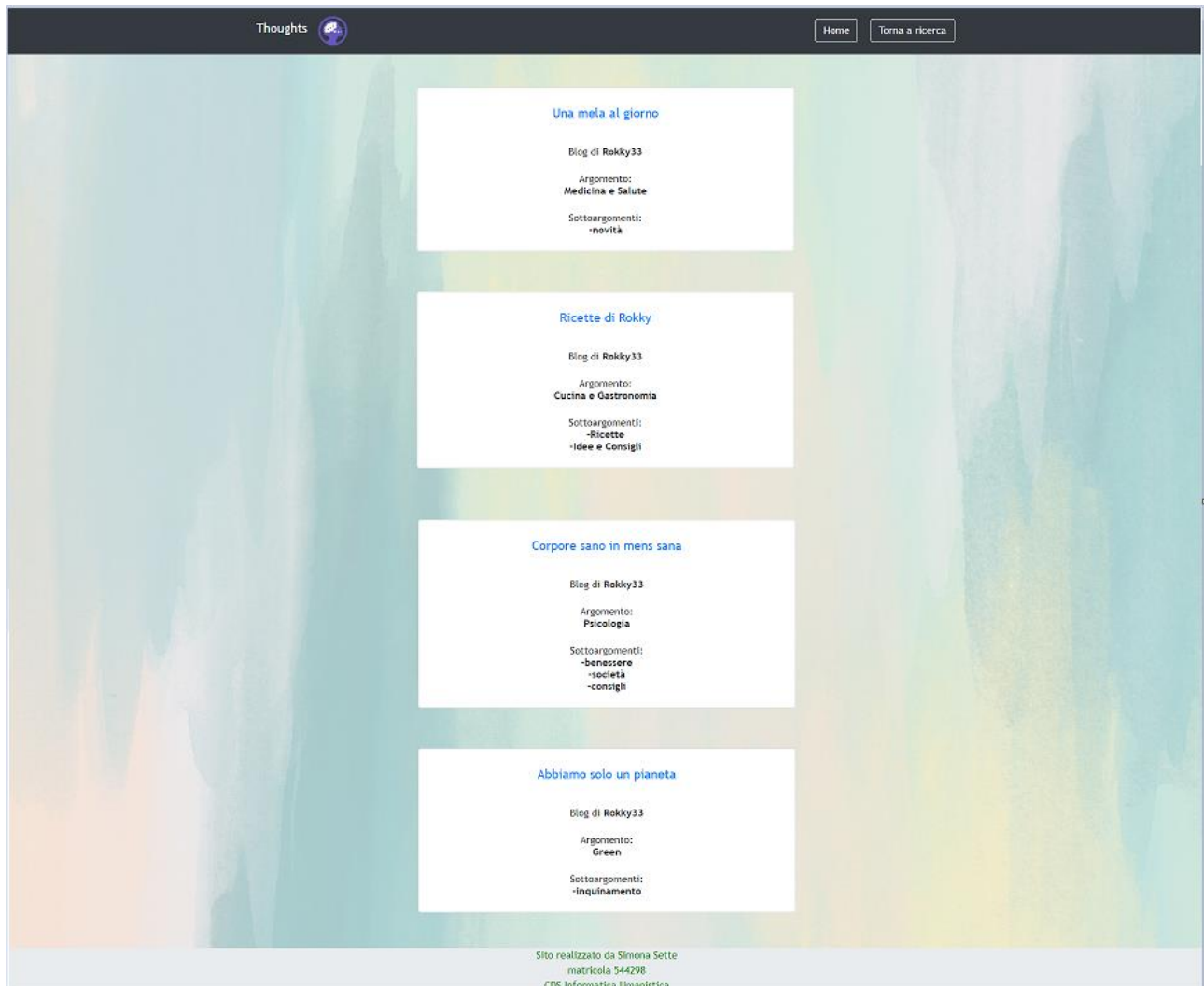
Qui l'utente può fare ricerche di blog basandosi su: argomenti, parola contenuta nel titolo del blog, nickname di un utente.

Fare una di queste ricerche porta a un reindirizzamento alla pagina Blogvision.php che mostra i risultati della ricerca

Blogvision.php

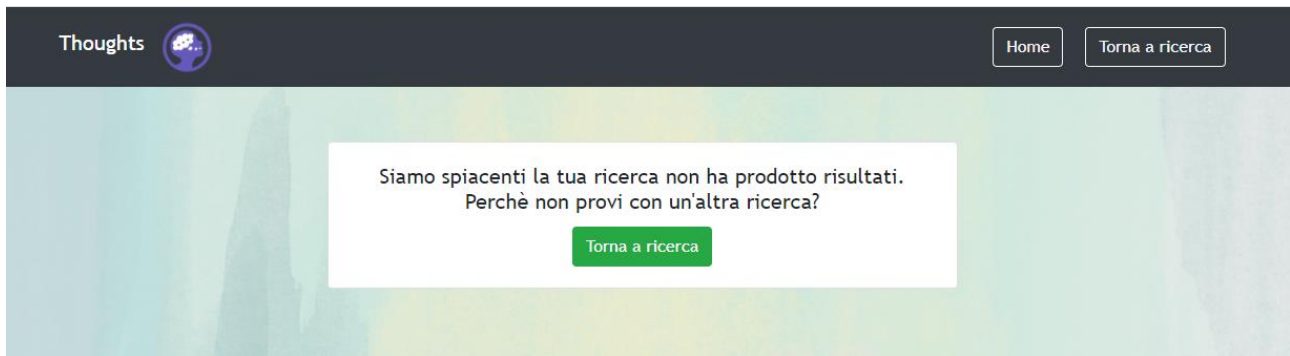
I blog la cui "anteprima" viene mostrata su questa pagina sono raggiungibili tramite il titolo/link al blog.

Risultato ricerca "o" nella ricerca dei blog tramite nickname autore:

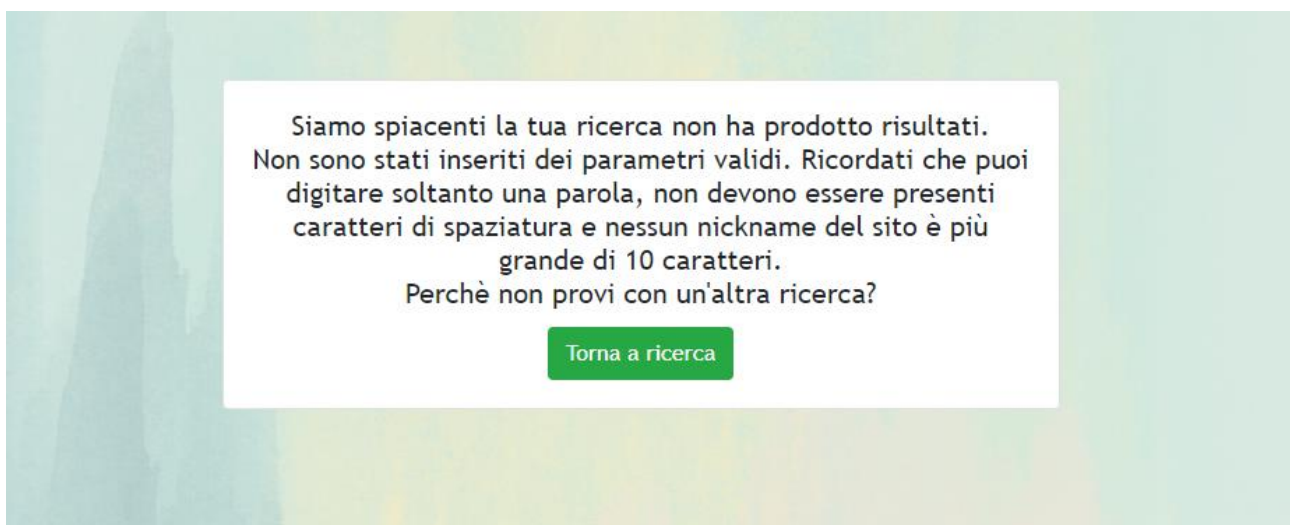


Quando la ricerca non porta risultati (per qualsiasi motivo) assieme al messaggio che lo comunica viene anche mostrato un bottone che riporta alla pagina di ricerca.

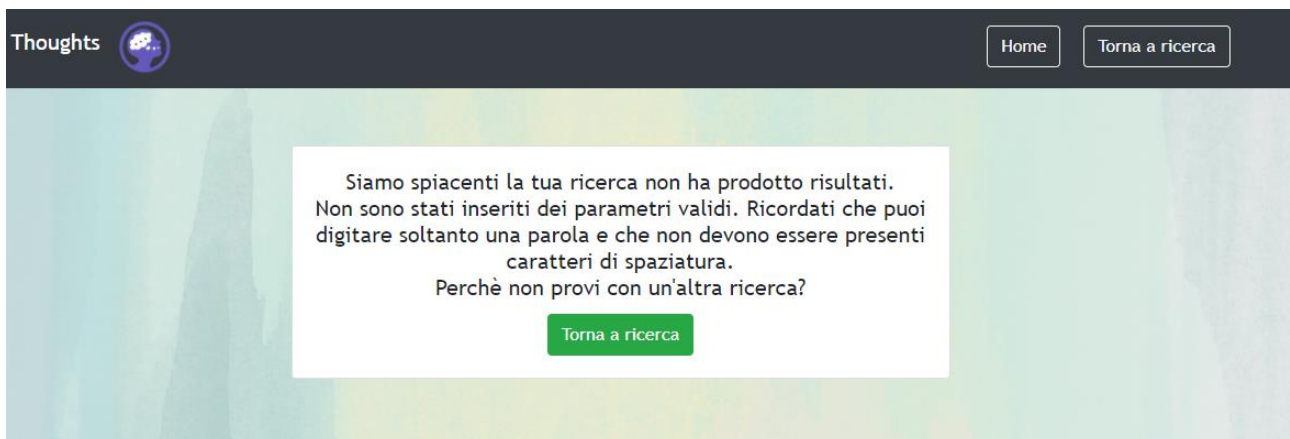
Nessun risultato della ricerca (nessun match):



Nessun risultato della ricerca (input del nickname non supera i controlli):



Nessun risultato della ricerca (input della parola non supera i controlli):



Documentazione delle funzioni PHP/ metodi jQuery usati:

Funzioni PHP:

- Iconv-strlen: <https://www.php.net/manual/en/function.iconv-strlen.php>
- begin-transaction: <https://www.php.net/manual/en/mysqli.begin-transaction.php>
- getimagesize: <https://www.php.net/manual/en/function.getimagesize.php>
- Microtime: <https://www.php.net/manual/en/function.microtime.php>
- Strtolower: <https://www.php.net/manual/en/function.strtolower.php>
- Pathinfo: <https://www.php.net/manual/en/function.pathinfo.php>
- Session-start: <https://www.php.net/manual/en/function.session-start.php>
- Exit: <https://www.php.net/manual/en/function.exit.php>
- Session-destroy: <https://www.php.net/manual/en/function.session-destroy.php>
- Json-encode: <https://www.php.net/manual/en/function.json-encode.php>
- Empty: <https://www.php.net/manual/en/function.empty.php>
- Isset: <https://www.php.net/manual/en/function.isset.php>
- Num-rows: <https://www.php.net/manual/en/mysqli-result.num-rows.php>
- Stripslashes: <https://www.php.net/manual/en/function.stripslashes.php>
- Esegue query: <https://www.php.net/manual/en/mysqli.query.php>
- Fetch_assoc: <https://www.php.net/manual/en/mysqli-result.fetch-assoc.php>
- Header: <https://www.php.net/manual/en/function.header.php>
- Date: <https://www.php.net/manual/en/function.date.php>

- Array: <https://www.php.net/manual/en/function.array.php>
- Preg-match: <https://www.php.net/manual/en/function.preg-match.php>
- Filter-var: <https://www.php.net/manual/en/function.filter-var.php>
- Move-uploaded-file: <https://www.php.net/manual/en/function.move-uploaded-file.php>
- Password-verify: <https://www.php.net/manual/en/function.password-verify.php>
- Password-hash: <https://www.php.net/manual/en/function.password-hash.php>

Prepared statement:

- Prepara una query→<https://www.php.net/manual/en/pdo.prepare.php>
- Bind dei parametri a una variabile→
<https://www.php.net/manual/en/pdostatement.bindparam.php>
- Esegue p.s.→
<https://www.php.net/manual/en/pdostatement.execute.php>
- Prende i risultati dell'esecuzione di una p.s→
<https://www.php.net/manual/en/mysqli-stmt.get-result.php>

Metodi jQuery:

- Prevent default: <https://api.jquery.com/event.preventdefault/>
- Html: <https://api.jquery.com/html/>
- Ajax: <https://api.jquery.com/jquery.ajax/>
- Find: <https://api.jquery.com/find/>
- Addclass: <https://api.jquery.com/addclass/>
- Click: <https://api.jquery.com/click/>
- Submit: <https://api.jquery.com/submit/>
- Val: <https://api.jquery.com/val/>
- Attr: <https://api.jquery.com/attr/>
- Each: <https://api.jquery.com/each/>
- Append: <https://api.jquery.com/append/>
- Done: <https://api.jquery.com/deferred.done/>
- Removeattr: <https://api.jquery.com/removeAttr/>

Documentazione degli elementi BOOTSTRAP usati:

- Selectpicker di bootstrap (jQuery plugin):
<https://developer.snapappointments.com/bootstrap-select/> (.selectpicker jquery)
- Cards :<https://getbootstrap.com/docs/4.0/components/card/>
- Carosello: <https://getbootstrap.com/docs/4.0/components/carousel/>
- Jumbotron: <https://getbootstrap.com/docs/4.0/components/jumbotron/>

- Button groups: <https://getbootstrap.com/docs/4.0/components/button-group/>
- Button: <https://getbootstrap.com/docs/4.0/components/buttons/>
- Forms: <https://getbootstrap.com/docs/4.0/components/forms/>
- Modal: <https://getbootstrap.com/docs/4.0/components/modal/> (.modals jquery)
- Navbar: <https://getbootstrap.com/docs/4.0/components/navbar/>
- Navs: <https://getbootstrap.com/docs/4.0/components/navs/>
- Containers: <https://getbootstrap.com/docs/4.0/layout/overview/>
- Sistema "griglia" di bootstrap: <https://getbootstrap.com/docs/4.0/layout/grid/>
- Spaziatura di bootstrap: <https://getbootstrap.com/docs/4.0/utilities/spacing/>

