Relazione progetto di Tecnologie Web

Andrea Delmastro
mat. n° 912954

a.a. 2021-2022

1 Introduzione

1.1 Tema

Il tema scelto è quello di una pubblicazione sportiva online chiamata *Vuvuzela*. Gli utenti sono suddivisi in utenti generici e `editor`. Il sito permette l'interazione con gli utenti tramite commenti, ricerche, filtri e liste di articoli personalizzate. Gli utenti `editor` hanno la possibilità di scrivere articoli, eliminarli e moderare i commenti.

1.2 Sezioni principali

Il sito espone quattro pagine:

1. **login**: permette la registrazione e l'accesso al sito.
2. **navigazione degli articoli**: si presenta come una singola pagina che viene aggiornata dinamicamente sulla base delle richieste effettuate dagli utenti. Un corpo centrale permette la navigazione e la lettura degli articoli. Ad esempio, lo stesso corpo centrare può contenere, a seconda del momento, la lista degli articoli preferiti da un utente, o la lista di tutti gli articoli taggati #*sumo*, o ancora la lista di tutti gli articoli che contengano *'Tennis'* nel titolo. Una sezione laterale permette di filtrare gli articoli, modificare il criterio di visualizzazione e mostra i profili di alcuni tra gli autori più prolifici del sito.
3. **scrittura di un articolo**: riservata agli utenti `editor`.
4. **moderazione dei commenti**: riservata agli utentu `editor` permette di accettare o rifiutare un commento. Solo i commenti accettati o in attesa di verifica vengono visualizzati nella sezione commenti di un articolo.

2 Funzionalità

2.1 Login, logout e registrazione

Login e registrazione sono entrambe gestite tramite una mediazione con ajax. Il server riceve le credenziali di accesso e ne verifica la validità rispondendo con determinati codici a seconda della condizione verificatasi (combinazione username/password non esistente, username già esistente ecc.). La procedura di login assegna i valori alle variabili di sessione (si veda la sezione Sessione per maggiori dettagli), mentre la procedura di logout libera il contenuto delle stesse variabili. Il logout è gestito tramite una chiamata ad uno script php non mediata da ajax. L'estrema semplicità dello script non richiede la segnalazione di condizione di errore all'utente. Ogni pagina permette l'accesso solo ad utenti registrati (distinguendo, in alcuni casi, in base al ruolo). In caso contrario, si viene rediretti alla pagina di login.

2.2 Contenuto e interazione dell'utente

Il sito permette di navigare, commentare e scrivere articoli a tema sportivo. Ogni articolo è associato a un autore, un titolo, un'immagine, un testo, una categoria, uno o più tag e una lista, eventualmente vuota di commenti. Un utente può leggere un articolo, commentarlo, *fare rumore* o aggiungerlo ad una delle sue liste (favoriti e da leggere). *Fare rumore* é l'equivalente di mettere

mi piace, ma produce un simpatico suono di vuvuzela ad ogni suo click. Le liste associate ad un utente sono private e visualizzabili solo dall'utente, mentre il *rumore* prodotto da ogni articolo viene utilizzato per stilare la classifica degli articoli più *rumorosi*. La sezione laterale mostra la possibilità di filtrare gli articoli in base a:

1. **lista di appartenenza**: articoli preferiti/articoli da leggere più tardi
2. **tag**: visualizza una lista con un misto tra i tag più popolari di sempre e i tag più popolari nell'ultimo periodo di tempo.
3. ***suono***: visualizza la lista degli articolo più *rumorosi*, con l'indicazione del rumore prodotto (numero di utenti che abbiano premuto `Make noise` in corrispondenza dell'articolo). La pressione sul pulsante associato a questa sezione permette di visualizzare gli articoli proposti nel corpo centrale (e di leggerli).
4. **Ricerca**: visualizza la lista degli articoli che contengono una certa parola o frase nel titolo.

e una lista degli utenti registrati come `editor` che abbiano prodotto almeno un articolo (in particolare, viene mostrata in oridine casuale una lista degli n autori più attivi che abbiano impostato l'immagine profilo). Si noti come un editor che abbia creato almeno un articolo è considerato un autore. La creazione dell'articolo prevede l'upload dell'immagine di copertina sul server.

3 Caratteristiche

3.1 Usabilità

Ad ogni azione utente corrisponde un feedback visisivo: ad esempio, nel caso della pressione di un pulsante di azione (come `Add to favourites`), il testo associato al pulsante viene modificato solo in caso in cui l'azione vada a buon fine. Ogni errore è segnalato tramite appositi messaggi di errore posizionati in corrispondenza dell'elemento che ha generato l'errore. A pulsanti con funzionalità differenti o con stati differenti (abilitato/disabilitato) sono associati stili differenti per permetterne la differenziazione.

3.2 Interazione/Animazioni

1. L'apertura di un articolo comporta il caricamento dal server remoto del testo associato all'articolo. Una volta che il testo è stato ottenuto, la pagina scorre tramite un'animazione jQuery fino a conincidere con il titolo dell'articolo.
2. La lista degli autori più attivi sul sito é visualizzata in uno carosello implementato tramite animazioni jQuery. In particolare, la pressione delle frecce poste ai due lati permette di transitare all'autore precedente o all'autore successivo tramite uno scorrimento *morbido*. Alcuni accorgimenti javascript sono stati implementati per permettere una corretta visualizzazione in un ambiente responsive.

3.3 Validazione dei dati di input

1. **Lato front-end**
   (a) Scrittura di un articolo: tutti i campi sono verificati essere non vuoti. Per il campo tags viene implementata una semplice regex che verifichi che la lista sia stata inserita nel modo corretto (ad esempio, un input corretto é il seguente: #sumo #japan #hakuho).

   ```
   /^(#\\w+ )*(#\\w+)? *\$/
   ```

   (b) Registrazione: tutti i campi sono verificati essere non vuoti. In generale, se i campi risultano non validi, viene segnalato all'utente tramite una differente colorazione. I pulsanti vengono abilitati solo in caso di input validi.
2. **Lato back-end**: tutti i parametri vengnono utilizzati nelle apposite query tramite i metodi offerti dalle classi php opportune per evitare attacchi XSS/SQL injection (si veda la sezione Sicurezza per maggiori dettagli).

3.4 Sicurezza

Lato server vengono evitati attacchi di SQL injection tramite l'utilizzo dei metodi offerti dalla classe PDO (`PDO->prepare()`, `PDO->bindValue()` e `PDO->execute()`). Ad esempio, la seguente è una sezione di codice che lega i parametri ricevuti dal client alla query:

```
function getFavourites(PDO $connection, int $from, int $num):
↪   bool|array {
    $statement = $connection->prepare(ARTICLE_QUERIES['favourites']);
    $statement->bindValue(':from', $from, PDO::PARAM_INT);
    /* [...] */
    $statement->execute();
    return $statement->fetchAll(PDO::FETCH_ASSOC);
}
```

Gli attacchi XSS vengono prevenuti tramite l'utilizzo della funzione `htmlspecialchars()` per ogni campo testuale immesso che venga poi inserito a livello HTML. Le password vengono memorizzate nel database sotto forma di hash utilizzando la funzione `password_hash()` della libreria standard php. La funzione `password_verify()` viene utilizzata per verificare che la password immessa corrisponda a quella memorizzata.

3.5 Sessioni

Le sessioni vengono utilizzate per memorizzare i dati associati all'utente che abbia effettuato l'accesso: `username`, `user_id` e `role`. Tutte le variabili associate alla sessione vengono reimpostate al logout. Il login rigenera il `session_id`.

3.6 Interrogazione del DB

L'interrogazione del DB avviene lato server tramite apposite query in cui vengono legati i valori inviati dal client. I valori vengono legati alle query tramite gli appositi metodi della classe PDO (si veda la sezione Sicurezza per maggiori dettagli).

3.7 Presentazione

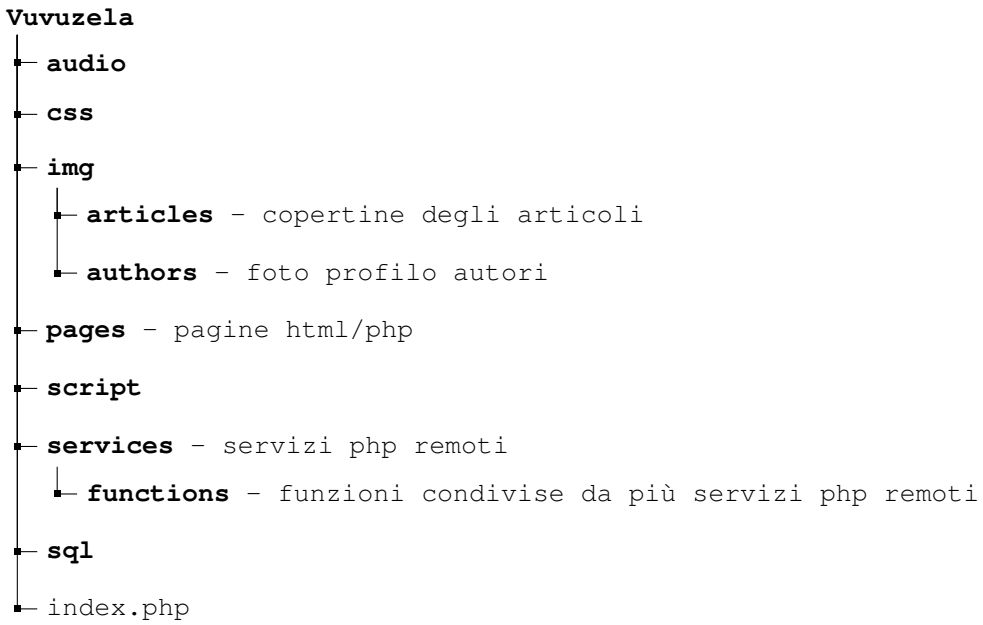
Il sito si presenta su un layout a due colonne. Gli elementi si distinguono sulla base della posizione (area centrale / sezione laterale), del font utilizzato (sans serif per i pulsanti, serif per i titoli e il testo degli articoli) e dei titoli associati a ciascuna sezione. Il layout è studiato per essere responsive tramite l'utilizzo di grid CSS (si veda la sezione Front-end per maggiori dettagli).

4 Front-end

4.1 Separazione presentazione/contenuto/comportamento

Ogni pagina è gestita utilizzando almeno tre file: un file di presentazione HTML o php, uno o più fogli di stile CSS e un file di script javascript. I file CSS sono organizzati in modo gerarchico per riutilizzare gli stili comuni a più pagine. Questa soluzione é particolarmente comoda in fase di sviluppo, può essere rivalutata in fase di distribuzione per limitare il numero di messaggi HTTP necessari a scaricare la pagina. Tutto il codice javascript é gestito in modalità unobtrusive.

4.2 Organizzazione file e cartelle di progetto



La cartella `sql` é stata aggiunta ai soli scopi della consegna per mostrare la struttura del database e alcuni valori fittizzi di esempio. Non é da considerarsi come parte dei file che verrebbero caricati sul server.

4.3 Soluzioni HTML/CSS/javascript degne di nota

4.3.1 Classi javascript

Ogni elemento grafico cui sia richiesta un'interazione complessa (creazione/distruzione) viene gestito da javascript tramite una particolare classe. L'utilizzo di un'approccio object-oriented permette di strutturare il codice in modo più semplice e chiaro. Ad esempio la seguente è la classe che definisce un commento:

```
class Comment {
    constructor(config, highlighted) { /* [...] */ }

    create($parent) {
        let html = /* HTML code */;
        this.highlighted ? $(html).prependTo($parent) :
        ↪  $(html).appendTo($parent);
    }

    delete() { $(`comment-${this.config['id']}`).remove(); }
}
```

Dove il metodo `create()` agisce sul DOM per creare gli elementi richiesti. Mentre la classe `CommentSection` conterrà una lista di riferimenti a `Comment`:

```
class CommentSection {
    constructor(articleID) {
        this.comments = [];
        /* [...] */
    }

    delete() {
        this.comments.forEach(comment => comment.delete());
        /* [...] */
    }
}
```

4.3.2 Layout CSS

L'organizzazione del layout fa massiccio utilizzo di grid e flexbox. L'utilizzo delle grid, in particolare, permette di modificare il layout in modo semplice in risposta al ridimensionamento della finestra. Nel seguente esempio il layout della pagina viene modificato da doppia colonna a singola colonna in relazione alla dimensione della finestra utilizzando una media query:

```
@media (min-width: 320px) {
    #home-content {
        grid-template-areas:
        'header'
        'articles'
        'aside'
        'footer';
        /* [...] */
    }

    /* [...] */
}

@media (min-width: 1025px) {
    #home-content {
        grid-template-areas:
        'header header'
        'articles aside'
        'footer footer';
        /* [...] */
    }

     /* [...] */
}
```

5 Back-end

5.1 Architettura generale classi/funzioni php

Ogni script php fa utilizzo di una o più funzioni. Le funzioni condivise (utilizzate da più script) vengono memorizzate in appositi file raggruppate per tipo. Si è cercato di limitare al massimo l'utilizzo di codice php nei file HTML. In particolare, quest'ultima soluzione é stata adottata esclusivamente per le `include()` di altre sezioni (header e footer) HTML e per la verifica delle condizioni di login.

5.2 Schema del DB

Il database è stato progettato appositamente per il progetto. Lo schema logico è il seguente:

article(id, author_id, title, description, text, date, img, category_name)

cateogry(name)

comment(id, text, date, verified, author_id, article_id)

favourite(user_id, article_id)

noise(user_id, article_id)

read_later(user_id, article_id)

tag(name)

tags(article_id, tag_name)

user(id, username, password, role, complete_name, img)

L'attributo `verified` della relazione `comment` viene utilizzato per indicare se il commento è stato accettato da parte di un utente moderatore (`editor`). L'attributo `password` della relazione `user` viene memorizzato come hash (si veda la sezione Sicurezza per maggiorni dettagli). Ogni altro campo è autoesplicativo e/o non adotta soluzioni di memorizzazione degne di nota.

5.3 Descrizione delle funzioni remote

A titolo di esempio (e per compattezza) viene fornita la descrizione della funzione remota di acquisizione degli articoli. La funzione viene chiamata in `POST` tramite una chiamata AJAX da parte del client. Nella chimata, vengono specificati:

1. `section`: articoli nuovi, articoli più *rumorosi*, articoli preferiti ecc.
2. `from`: numero di ordine del primo articolo da scaricare
3. `num`: numero di articoli da scaricare
4. `tag`: eventuale tag con cui filtrare gli articoli

L'output prodotto dalla funzione remota, in formato json, ha questa struttura:

```
{
    "status": "success",
    "data": [
      {
        "id": 3,
        "title": "Manchester is blue",
        "description": "Is it time to change what we have always
        ↪  thought about football in Manchester?",
        "author": "Andrea Delmastro",
        "date": "January 06, 2022",
        "img": "city.jpg",
        "category": "Football",
        "noise": 1,
        "noised": 0,
        "favourite": 0,
        "read_later": 0
      },
    ]
}
```

Le funzioni di callback lato javascript aggiornano la pagina con i nuovi articoli richiesti.
Le condizioni di errore che si possono verificare sono:

1. Accesso non eseguito: l'utente viene reindirizzato verso la pagina di login.
2. Richiesta mal formulata/errore nella query: lo stato della risposta é impostato a "error" e il campo "errorMsg" é impostato con il messaggio di errore corrispondente. L'errore è segnalato in corrispondenza dell'elemento che lo ha generato tramite un elemento grafico apposito.