FILE PHP :

File confin.php :

Il codice si connette al database MySQL utilizzando le informazioni di accesso specificate e verifica se la connessione è stata stabilita correttamente.
Viene creato un nuovo oggetto `MySQLi` utilizzando le informazioni di connessione fornite.

File db.php :

Il codice fornisce funzioni per eseguire operazioni di base su un database MySQL, come selezione, inserimento, aggiornamento ed eliminazione di record. Include anche funzioni specifiche per recuperare post pubblicati, post per ID argomento e post per ricerca.

Ecco cosa fanno queste funzioni:

1. `executeQuery($sql, $data)` (linea 13-22): Questa funzione prepara e esegue una query SQL utilizzando i parametri forniti. Prepara la query utilizzando il metodo `prepare()` dell'oggetto connessione al database (`$conn`), imposta i valori dei parametri utilizzando `bind_param()`, esegue la query con `execute()` e restituisce lo statement (`$stmt`) risultante.

2. `selectAll($table, $conditions = [])` (linea 25-47): Questa funzione seleziona tutte le righe dalla tabella specificata nel primo parametro (`$table`). Se sono presenti condizioni specificate nel secondo parametro (`$conditions`), viene generata una clausola WHERE nella query per filtrare i risultati. Restituisce un array associativo contenente tutti i record ottenuti dalla query.

3. `selectOne($table, $conditions)` (linea 50-72): Questa funzione seleziona una singola riga dalla tabella specificata nel primo parametro (`$table`). Vengono utilizzate le condizioni specificate nel secondo parametro (`$conditions`) per filtrare i risultati. Restituisce un array associativo che rappresenta il record selezionato.

4. `create($table, $data)` (linea 75-91): Questa funzione crea un nuovo record nella tabella specificata nel primo parametro (`$table`). I dati da inserire vengono forniti come un array associativo nel secondo parametro (`$data`). Restituisce l'ID del record appena creato.

5. `update($table, $id, $data)` (linea 94-114): Questa funzione aggiorna un record esistente nella tabella specificata nel primo parametro (`$table`). L'ID del record da aggiornare viene fornito nel secondo parametro (`$id`), mentre i nuovi dati da aggiornare vengono forniti come un array associativo nel terzo parametro (`$data`). Restituisce il numero di righe interessate dall'operazione di aggiornamento.

6. `delete($table, $id)` (linea 117-125): Questa funzione elimina un record dalla tabella specificata nel primo parametro (`$table`). L'ID del record da eliminare viene fornito nel secondo parametro (`$id`). Restituisce il numero di righe interessate dall'operazione di eliminazione.

7. `getPublishedPosts()` (linea 128-141): Questa funzione recupera tutti i post pubblicati dal database, inclusi i nomi utente degli autori corrispondenti. Restituisce un array associativo contenente i record dei post pubblicati.

8. `getPostsByTopicId($topic_id)` (linea 144-157): Questa funzione recupera i post pubblicati dal database che corrispondono all'ID dell'argomento fornito (`$topic_id`). Include anche i nomi utente degli autori corrispondenti. Restituisce un array associativo contenente i record dei post corrispondenti all'argomento.

9. `searchPosts($term)` (linea 160-174): Questa funzione effettua una ricerca nel database per post pubblicati che corrispondono al termine di ricerca fornito (`$term`). La ricerca viene eseguita sui titoli e sui contenuti dei post. Restituisce un array associativo contenente i record dei post corrispondenti alla ricerca.

File formError.php :

Il codice controlla se ci sono errori nell'array `$errors` e, se presenti, li visualizza in un blocco HTML separati da elementi di lista `<li>`. Questo è utilizzato per visualizzare messaggi di errore all'utente in una pagina web.

Mostra un blocco di codice che viene eseguito solo se l'array `$errors` contiene almeno un elemento. Ecco cosa fa il codice:

1. Verifica se l'array `$errors` contiene almeno un elemento utilizzando la funzione `count($errors)`. Se l'array contiene elementi, l'espressione `count($errors) > 0` sarà valutata come `true`.

2. Se l'array `$errors` contiene elementi, viene visualizzato un blocco HTML con la classe CSS "msg error". Questo blocco viene utilizzato per mostrare messaggi di errore all'utente.

3. All'interno del blocco HTML, viene eseguito un ciclo `foreach` su ogni elemento dell'array `$errors`. Per ogni elemento, viene visualizzato un elemento `<li>` che contiene il valore dell'errore (`$error`) utilizzando l'istruzione `echo`.

File logout.php :

Il codice esegue l'operazione di logout distruggendo la sessione utente corrente e reindirizzando l'utente a una pagina specificata dopo il logout.

Esegue diverse azioni relative alla sessione utente. Ecco cosa fa il codice:

1. Avvia la sessione utilizzando `session_start()`. Questa funzione inizializza o ripristina una sessione utente esistente.

2. Vengono utilizzate le istruzioni `unset()` per rimuovere le variabili di sessione specificate. Nel codice fornito, le variabili di sessione rimosse sono `id`, `username`, `admin`, `message` e `type`. Ciò significa che vengono cancellati i valori associati a queste variabili nella sessione corrente.

3. Viene chiamata la funzione `session_destroy()` per distruggere completamente la sessione. Questo elimina tutti i dati associati alla sessione corrente.

4. Infine, viene utilizzata l'istruzione `header()` per reindirizzare l'utente a una nuova pagina.

File message.php :
Il codice controlla se è presente un messaggio nella variabile di sessione `$_SESSION['message']` e, se presente, lo visualizza in un blocco HTML. Dopo la visualizzazione del messaggio, le variabili di sessione associate vengono cancellate per assicurarsi che il messaggio venga mostrato solo una volta.
Mostra un blocco di codice che viene eseguito solo se la variabile di sessione `$_SESSION['message']` è impostata. Ecco cosa fa il codice:

1. Verifica se la variabile di sessione `$_SESSION['message']` è impostata utilizzando `isset($_SESSION['message'])`. Se la variabile di sessione è impostata, l'espressione `isset($_SESSION['message'])` sarà valutata come `true`.

2. Se la variabile di sessione `$_SESSION['message']` è impostata, viene visualizzato un blocco HTML con una classe CSS determinata dal valore della variabile di sessione `$_SESSION['type']`. Questo blocco viene utilizzato per mostrare un messaggio all'utente, ad esempio un messaggio di successo, un messaggio di errore.

3. All'interno del blocco HTML, viene visualizzato un elemento di lista `<li>` che contiene il valore della variabile di sessione `$_SESSION['message']` utilizzando l'istruzione `echo`.

4. Successivamente, le variabili di sessione `$_SESSION['message']` e `$_SESSION['type']` vengono cancellate utilizzando l'istruzione `unset()`. Ciò significa che i valori associati a queste variabili nella sessione corrente vengono rimossi.

File middleware.php :
Il codice PHP fornito definisce tre funzioni per il controllo dell'accesso utente in base al ruolo dell'utente. Queste funzioni sono utilizzate per controllare l'accesso degli utenti in base al ruolo (utente normale o amministratore) e al fatto che abbiano effettuato l'accesso o meno. Se un utente non soddisfa i requisiti di accesso, viene reindirizzato a una pagina specifica con un messaggio di errore. Ecco cosa fanno le funzioni:

1. `usersOnly($redirect = 'php/register.php')`: Questa funzione controlla se l'utente ha effettuato l'accesso. Se l'utente non ha effettuato l'accesso (la variabile di sessione `$_SESSION['id']` è vuota), viene impostato un messaggio di errore e viene reindirizzato l'utente alla pagina specificata nel parametro `$redirect`. Di default, se il parametro `$redirect` non viene specificato, l'utente viene reindirizzato a 'php/register.php'. La funzione `exit(0)` viene utilizzata per terminare immediatamente l'esecuzione dello script dopo il reindirizzamento.

2. `adminOnly($redirect = 'php/index.php')`: Questa funzione controlla se l'utente ha effettuato l'accesso come amministratore. Verifica se la variabile di sessione `$_SESSION['id']` o `$_SESSION['admin']` sono vuote. Se una di queste variabili è vuota, viene impostato un messaggio di errore e l'utente viene reindirizzato alla pagina specificata nel parametro `$redirect`. Di default, se il parametro `$redirect` non viene specificato, l'utente viene reindirizzato a 'php/index.php'. Anche in questo caso, la funzione `exit(0)` viene utilizzata per terminare immediatamente l'esecuzione dello script dopo il reindirizzamento.

3. `guestOnly($redirect = 'php/index.php')`: Questa funzione controlla se l'utente è un ospite (non ha effettuato l'accesso). Verifica se la variabile di sessione `$_SESSION['id']` è impostata. Se la

variabile di sessione è impostata, significa che l'utente ha già effettuato l'accesso e viene reindirizzato alla pagina specificata nel parametro `$redirect`. Di default, se il parametro `$redirect` non viene specificato, l'utente viene reindirizzato a 'php/index.php'. Anche in questo caso, la funzione `exit(0)` viene utilizzata per terminare immediatamente l'esecuzione dello script dopo il reindirizzamento.

File path.php :

Il codice definisce due costanti: `ROOT_PATH`, che rappresenta il percorso assoluto della directory corrente, e `BASE_URL`, che rappresenta l'URL base del sito. Queste costanti vengono utilizzate per creare percorsi e URL assoluti all'interno del progetto.

File posts.php :

Il codice gestisce l'aggiunta, la modifica e l'eliminazione di post nel database. Gestisce lo stato di pubblicazione dei post e i messaggi di notifica per le operazioni eseguite. Ecco cosa fa il codice:

1. Viene definita la variabile `$table` con il valore "posts". Questo rappresenta il nome della tabella del database a cui si riferiscono i post.

2. Viene eseguita la query `selectAll('topics')` per recuperare tutti gli argomenti disponibili e viene assegnato il risultato alla variabile `$topics`.

3. Viene eseguita la query `selectAll($table)` per recuperare tutti i post dalla tabella specificata e viene assegnato il risultato alla variabile `$posts`.

4. Vengono definite le variabili `$errors`, `$id`, `$title`, `$body`, `$topic_id` e `$published` con valori predefiniti. Queste variabili vengono utilizzate per la gestione degli errori e per mantenere i valori dei campi del post durante l'aggiunta o la modifica.

5. Se è presente un parametro GET `del_id`, viene eseguita la funzione `adminOnly()` per verificare se l'utente è un amministratore. Successivamente, viene eseguita la funzione `delete($table, $_GET['del_id'])` per eliminare il post con l'ID specificato. Viene impostato un messaggio di successo nella variabile di sessione `$_SESSION['message']` e si viene reindirizzati alla pagina "php/index-post.php".

6. Se sono presenti i parametri GET `published` e `p_id`, viene eseguita la funzione `adminOnly()` per verificare se l'utente è un amministratore. Successivamente, viene eseguita la funzione `update($table, $p_id, ['published' => $published])` per aggiornare lo stato di pubblicazione del post corrispondente all'ID specificato. Viene impostato un messaggio di successo nella variabile di sessione `$_SESSION['message']` e si viene reindirizzati alla pagina "php/index-post.php".

7. Se è presente il parametro GET `id`, viene eseguita la funzione `selectOne($table, ['id' => $_GET['id']])` per recuperare il post corrispondente all'ID specificato. Vengono assegnati i valori dei campi del post alle variabili corrispondenti (`$id`, `$title`, `$body`, `$topic_id`, `$published`).

8. Se viene inviato il form con il nome "add-post", viene eseguito il codice per l'aggiunta di un nuovo post. Viene eseguita la funzione `adminOnly()` per verificare se l'utente è un amministratore. Viene eseguita la funzione `validatePost($_POST)` per verificare la validità dei dati inviati nel form. Viene controllato se è stata caricata un'immagine e, in caso positivo, viene

spostata nella directory di destinazione. Se non sono presenti errori di validazione, i dati del post vengono salvati nel database utilizzando la funzione `create($table, $_POST)`. Viene impostato un messaggio di successo nella variabile di sessione `$_SESSION['message']` e si viene reindirizzati alla pagina "php/index-post.php".

9. Se viene inviato il form con il nome "update-post", viene eseguito il codice per la modifica di un post esistente. Vengono eseguite le stesse operazioni di validazione dell'aggiunta di un post, ma utilizzando la funzione `update($table, $id, $_POST)` per aggiornare i dati del post nel database. Viene impostato un messaggio di successo nella variabile di sessione `$_SESSION['message']` e si viene reindirizzati alla pagina "php/index-post.php".

File process_feedback.php :

Il codice gestisce l'aggiunta di commenti di valutazione nel database tramite una richiesta POST. Verifica se sono presenti errori, esegue l'inserimento nel database e restituisce una risposta JSON indicante lo stato dell'operazione (successo o errore). Ecco cosa fa il codice:

1. Viene definito un array vuoto `$errors` per gestire eventuali errori durante il processo di aggiunta del commento.

2. Se il form di invio del commento viene inviato (i campi "name", "rating" e "comment" non sono vuoti), viene chiamata la funzione `createFeedback($_POST)` per creare il commento.

3. La funzione `createFeedback($request_values)` viene definita per gestire il processo di creazione del commento nel database. Vengono passati i valori della richiesta POST come argomento.

4. Nella funzione `createFeedback()`, vengono inizializzate le variabili `$name`, `$rating` e `$comment` utilizzando i valori dalla richiesta POST.

5. Viene eseguito un controllo per verificare se ci sono errori. In questo caso, l'array `$errors` dovrebbe essere vuoto.

6. Se non ci sono errori, viene eseguita una query di inserimento nel database per inserire il commento nella tabella "valutazioni". La query utilizza i valori di `$name`, `$rating` e `$comment`.

7. Viene restituito un messaggio di successo tramite la variabile `$message` e viene creata una variabile `$status` che contiene un array con le chiavi "error" e "message".

8. Se ci sono errori, viene creato un messaggio di errore tramite la variabile `$message` e viene creata una variabile `$status` che contiene un array con le chiavi "error" e "message".

9. Viene utilizzata la funzione `json_encode()` per convertire l'array `$status` in una stringa JSON.

10. Viene stampata la stringa JSON per essere utilizzata come risposta nell'applicazione client

File show_feedback.php :
Il codice recupera i commenti di valutazione dal database e li mostra nell'output HTML. Viene eseguita una query per ottenere i commenti, quindi viene generato l'HTML per visualizzare ogni commento, comprensivo di contenuto, valutazione e autore. Infine, l'HTML viene stampato nell'output per essere visualizzato nell'applicazione client. Ecco cosa fa il codice:

1. Viene eseguita una query per selezionare tutti i campi "name", "valutazione" e "commento" dalla tabella "valutazioni". Il risultato della query viene assegnato alla variabile `$commentsResult`.

2. Viene inizializzata una stringa vuota `$commentHTML` che conterrà l'HTML per mostrare i commenti.

3. Viene eseguito un loop while per estrarre i commenti dal risultato della query. Per ogni commento, vengono concatenati gli elementi HTML necessari per visualizzare il contenuto del commento, la valutazione e l'autore. Questi elementi vengono aggiunti alla stringa `$commentHTML`.

4. Se non ci sono commenti nel database (la variabile `$commentHTML` è vuota), viene aggiunto un messaggio HTML per indicare che non sono ancora stati pubblicati commenti.

5. Viene stampata la stringa `$commentHTML`, che rappresenta l'output HTML per visualizzare i commenti.

File topics.php :
Il codice gestisce l'aggiunta, la modifica e l'eliminazione dei topic nel database. Contiene anche la validazione dei dati del topic e le verifiche di accesso per le operazioni riservate agli amministratori. Ecco cosa fa il codice:

1. Viene definita la variabile `$table` con il valore "topics". Questo rappresenta il nome della tabella del database che contiene i topic.

2. Viene inizializzato l'array `$errors` per gestire eventuali errori durante l'aggiunta o la modifica dei topic.

3. Viene eseguita la query `selectAll($table)` per recuperare tutti i topic dalla tabella specificata e il risultato viene assegnato alla variabile `$topics`.

4. Se viene inviato il form con il nome "add-topic", viene eseguito il codice per l'aggiunta di un nuovo topic. Viene eseguita la funzione `adminOnly()` per verificare se l'utente è un amministratore. Viene eseguita la funzione `validateTopic($_POST)` per verificare la validità dei dati inviati nel form. Se non ci sono errori di validazione, i dati del topic vengono salvati nel database utilizzando la funzione `create('topics', $_POST)`. Viene impostato un messaggio di successo nella variabile di sessione `$_SESSION['message']` e si viene reindirizzati alla pagina "php/index-topic.php".

5. Se è presente il parametro GET `id`, viene eseguita la funzione `selectOne($table, ['id' => $id])` per recuperare il topic corrispondente all'ID specificato. Vengono assegnati i valori dei campi del topic alle variabili corrispondenti (`$id`, `$name`).

6. Se è presente il parametro GET `del_id`, viene eseguita la funzione `adminOnly()` per verificare se l'utente è un amministratore. Viene eseguita la funzione `delete($table, $id)` per eliminare il topic corrispondente all'ID specificato. Viene impostato un messaggio di successo nella variabile di sessione `$_SESSION['message']` e si viene reindirizzati alla pagina "php/index-topic.php".

7. Se viene inviato il form con il nome "update-topic", viene eseguito il codice per la modifica di un topic esistente. Viene eseguita la funzione `adminOnly()` per verificare se l'utente è un amministratore. Viene eseguita la funzione `validateTopic($_POST)` per verificare la validità dei dati inviati nel form. Se non ci sono errori di validazione, i dati del topic vengono aggiornati nel database utilizzando la funzione `update($table, $id, $_POST)`. Viene impostato un messaggio di successo nella variabile di sessione `$_SESSION['message']` e si viene reindirizzati alla pagina "php/index-topic.php".

File users.php :

Il codice PHP fornito gestisce le operazioni di registrazione, accesso e gestione degli utenti nel sistema. Ecco cosa fa il codice:

1. Viene definita la variabile '$table' con il valore "users". Questo rappresenta il nome della tabella del database che contiene le informazioni degli utenti.

2. Viene inizializzato l'array '$errors' per gestire eventuali errori durante il processo di registrazione, accesso o gestione degli utenti.

3. Viene definita una serie di variabili '($id, $username, $admin, $email, $password, $passwordConf)' che rappresentano i campi relativi agli utenti e vengono inizializzate con valori vuoti.

4. La funzione 'loginUser($user)' viene definita per impostare le variabili di sessione per l'utente loggato e reindirizzare l'utente alla pagina corretta. Se l'utente è un amministratore, viene reindirizzato alla pagina "php/dashboard.php", altrimenti alla pagina "php/index.php".

5. Se viene inviato il form con il nome "register-btn" o "create-admin", viene eseguito il codice per la registrazione di un nuovo utente. Viene eseguita la funzione 'validateUsers($_POST)' per verificare la validità dei dati inviati nel form. Se non ci sono errori di validazione, i dati dell'utente vengono salvati nel database utilizzando la funzione 'create($table, $_POST)'. Se l'utente è un amministratore, viene impostato un messaggio di successo nella variabile di sessione '$_SESSION['message']' e si viene reindirizzati alla pagina "php/dashboard.php". Altrimenti, viene effettuato il login automatico dell'utente utilizzando la funzione loginUser($user).

6. Se è presente il parametro 'GET id', viene eseguita la funzione 'selectOne($table, ['id' => $id])' per recuperare le informazioni dell'utente corrispondente all'ID specificato. Vengono assegnati i valori dei campi dell'utente alle variabili corrispondenti '($id, $username, $admin, $email)'.

7. Se viene inviato il form con il nome "update-user", viene eseguito il codice per la modifica delle informazioni di un utente esistente. Viene eseguita la funzione 'adminOnly()' per verificare se l'utente è un amministratore. Viene eseguita la funzione 'validateUsers($_POST)' per verificare la validità dei dati inviati nel form. Se non ci sono errori di validazione, i dati dell'utente vengono

aggiornati nel database utilizzando la funzione ‘update($table, $id, $_POST)’. Viene impostato un messaggio di successo nella variabile di sessione ‘$_SESSION['message']’ e si viene reindirizzati alla pagina "php/index-user.php".

8. Se viene inviato il form con il nome "login-btn", viene eseguito il codice per il processo di accesso. Viene eseguita la funzione ‘validateLogin($_POST)’ per verificare la validità dei dati inviati nel form. Se non ci sono errori di validazione, viene effettuato il controllo delle credenziali dell'utente. Se le credenziali sono corrette, viene chiamata la funzione ‘loginUser($user)’ per effettuare il login dell'utente.

9. Se è presente il parametro ‘GET del_id’, viene eseguita la funzione ‘adminOnly()’ per verificare se l'utente è un amministratore. Viene eseguita la funzione ‘delete($table, $_GET['del_id'])’ per eliminare l'utente corrispondente all'ID specificato. Viene impostato un messaggio di successo nella variabile di sessione ‘$_SESSION['message']’ e si viene reindirizzati alla pagina "php/index-user.php”.

File validatePost.php :

La funzione "validatePost" controlla la validità dei dati di un post e restituisce un array contenente eventuali errori rilevati durante la validazione.
Definisce una funzione chiamata "validatePost" che prende come parametro un array associativo contenente i dati di un post. La funzione verifica se i campi obbligatori del post, come il titolo, il corpo e l'ID del topic, sono vuoti. Se uno o più di questi campi sono vuoti, l'errore corrispondente viene aggiunto all'array degli errori.

Successivamente, la funzione controlla se esiste già un post con lo stesso titolo nel database. Se un post esistente viene trovato e l'azione corrente è una modifica (indicata dall'isset di 'update-post') e l'ID del post esistente è diverso dall'ID del post corrente, viene aggiunto un errore che segnala la duplicazione del titolo. Se invece l'azione corrente è un'aggiunta di un nuovo post (indicata dall'isset di 'add-post'), viene aggiunto un errore che segnala la duplicazione del titolo.

File validateTopic.php :

La funzione "validateTopic" controlla la validità dei dati di un topic e restituisce un array contenente eventuali errori rilevati durante la validazione.
Definisce una funzione chiamata "validateTopic" che prende come parametro un array associativo contenente i dati di un topic. La funzione verifica se il campo obbligatorio del nome del topic è vuoto. Se il campo è vuoto, viene aggiunto un errore corrispondente all'array degli errori.

Successivamente, la funzione controlla se esiste già un topic con lo stesso nome nel database. Se un topic esistente viene trovato e l'azione corrente è una modifica (indicata dall'isset di 'update-topic') e l'ID del topic esistente è diverso dall'ID del topic corrente, viene aggiunto un errore che segnala la duplicazione del nome. Se invece l'azione corrente è un'aggiunta di un nuovo topic (indicata dall'isset di 'add-topic'), viene aggiunto un errore che segnala la duplicazione del nome.

File validateUser.php :

La funzione "validateUsers" verifica la validità dei dati di registrazione degli utenti, mentre la funzione "validateLogin" verifica la validità dei dati di accesso degli utenti. Entrambe le funzioni restituiscono un array contenente eventuali errori rilevati durante la validazione.

Definisce due funzioni: "validateUsers" e "validateLogin".

La funzione "validateUsers" prende come parametro un array associativo contenente i dati di un utente. La funzione verifica se i campi obbligatori come username, email e password sono vuoti. Se uno qualsiasi di questi campi è vuoto, viene aggiunto un errore corrispondente all'array degli errori. Inoltre, la funzione controlla se la conferma della password corrisponde alla password inserita. Se le password non coincidono, viene aggiunto un errore.

Successivamente, la funzione controlla se esiste già un utente con lo stesso indirizzo email nel database. Se un utente esistente viene trovato, viene aggiunto un errore che segnala la duplicazione dell'email.

La funzione "validateLogin" prende come parametro un array associativo contenente i dati di accesso di un utente. La funzione verifica se i campi obbligatori come username e password sono vuoti. Se uno qualsiasi di questi campi è vuoto, viene aggiunto un errore corrispondente all'array degli errori.

FILE JS:

File editor.js :

Il codice configura e inizializza l'editor TinyMCE con determinate opzioni per la formattazione del testo e la visualizzazione degli strumenti, consentendo agli utenti di formattare il testo e inserire contenuti in un'area di testo specifica.

L'editor viene configurato con diverse opzioni:

- 'selector' specifica l'elemento HTML su cui verrà applicato l'editor.
- 'height' specifica l'altezza dell'area di testo dell'editor.
- 'toolbar' definisce quali pulsanti di formattazione e strumenti saranno visualizzati nella barra degli strumenti dell'editor. I pulsanti specificati includono opzioni per annullare/ripristinare azioni, selezionare formati di testo, applicare grassetto, corsivo, colore di sfondo, allineare il testo e creare elenchi puntati o numerati.
- 'menubar' specifica quali menu saranno visualizzati nella barra del menu dell'editor. I menu specificati includono preferiti, file, modifica, visualizza, inserisci, formato, strumenti, tabella e aiuto.
- 'content_css' specifica il percorso al file CSS che verrà utilizzato per applicare lo stile al contenuto dell'editor.

In sintesi, il codice configura e inizializza l'editor TinyMCE con determinate opzioni per la formattazione del testo e la visualizzazione degli strumenti, consentendo agli utenti di formattare il testo e inserire contenuti in un'area di testo specifica.

File feedback.js :
Il codice gestisce l'invio dei feedback attraverso una richiesta AJAX e visualizza i feedback aggiornati senza dover ricaricare l'intera pagina.

Il codice fornito utilizza la libreria jQuery per gestire l'invio e la visualizzazione dei feedback attraverso una richiesta AJAX.

Il codice è suddiviso in due parti:

1. All'interno della funzione `$(document).ready()`, viene definita l'azione da eseguire quando il documento è completamente caricato. Viene richiamata la funzione `showFeedback()` per visualizzare i feedback già presenti nella pagina.

Inoltre, viene impostato l'evento di submit del form con id "feedback_form" per evitare l'invio predefinito del modulo e gestire l'invio del feedback tramite AJAX. Quando viene inviato il modulo, i dati vengono serializzati e inviati al file "process_feedback.php" tramite una richiesta POST. Il tipo di dati atteso è JSON. In caso di successo, la risposta viene gestita nella funzione di callback success, che verifica se ci sono errori nella risposta. Se non ci sono errori, il form viene resettato, viene visualizzato un messaggio di conferma e viene richiamata nuovamente la funzione `showFeedback()` per aggiornare la visualizzazione dei feedback. Se ci sono errori, viene visualizzato un messaggio di errore.

2. La funzione `showFeedback()` viene chiamata all'avvio del documento e successivamente all'invio di un feedback. Utilizza una richiesta AJAX per inviare i dati del form al file "show_feedback.php" tramite una richiesta POST. La risposta viene inserita nell'elemento con id "show_feedback" per visualizzare i feedback aggiornati.

File script.js :
Il codice inizializza uno slider di post utilizzando il plugin Slick, consentendo di visualizzare più post contemporaneamente e di scorrere tra di essi in modo automatico o tramite pulsanti di navigazione.

Il codice fornito utilizza la libreria jQuery e il plugin Slick per creare uno slider di post.

All'interno della funzione `$(document).ready()`, viene chiamato il metodo `slick()` sul selettore `.post-wrapper`. Questo metodo inizializza lo slider e specifica le opzioni di configurazione. Le opzioni includono:

- `slidesToShow: 3`: specifica il numero di slide da mostrare contemporaneamente nello slider.
- `slidesToScroll: 1`: specifica il numero di slide da scorrere alla volta quando si naviga nello slider.
- `autoplay: true`: abilita la riproduzione automatica dello slider.
- `autoplaySpeed: 2000`: specifica la velocità di transizione tra le slide durante la riproduzione automatica.
- `nextArrow: $('.next')` e `prevArrow: $('.prev')`: specificano gli elementi HTML che fungono da pulsanti di navigazione successiva e precedente nello slider.