Diario di lavoro

Luogo	Trevano
Data	26.09.2019

Lavori svolti

Oggi ho contiunuato con l'implementazione della pagina principale. Inizialmente ho messo a posto un piccolo problema che ho notato nella pagina principale: lo zoom sulla mappa non sempre eseguiva l'animazione e l'ho risolto cambiando il valore che esso doveva raggiungere (io richiedevo uno zoom di 7 quando il minimo era 9). Dopodiché ho iniziato a implementare il frontend della pagina di login, questo mi serviva dato che è da essa che si può passare a quella di registrazione. Grazie al framework, Material Design Bootstrap, che sto utilizzando l'operazione è stata piuttosto veloce. Fatto ciò ho creato anche il frontend della pagina di registrazione con lo stesso metodo e stile di quella di login. Ho poi continuato facendo i controlli backend dei dati inseriti, essi verificano inizialmente che il metodo di connessione sia corretto (post), poi che tutti i campi siano stati compilati ed infine richiama una funzione che elimina da essi possibili caratteri speciali e malevoli.
La classe è la seguente:

```
class InputManager
{
    public function checkInput($input){
        $input = trim($input);
        $input = stripslashes($input);
        $input = htmlspecialchars($input);
        return $input;
    }
}
```

Una volta controllati tutti i valori essi vengono controllati con degli if che ne verificano la lunghezza.

```
$im = new InputManager();
$firstname = filter_var($im->checkInput($_POST['firstname']), FILTER_SANITIZE_STRING);
$lastname = filter_var($im->checkInput($_POST['lastname']), FILTER_SANITIZE_STRING);
$username = filter_var($im->checkInput($_POST['username']), FILTER_SANITIZE_STRING);
$email = filter_var($im->checkInput($_POST['email']), FILTER_SANITIZE_EMAIL);
$password = filter_var($im->checkInput($_POST['password']), FILTER_SANITIZE_STRING);
$repassword =filter_var($im->checkInput($_POST['repassword']), FILTER_SANITIZE_STRING);

if(!(strlen($firstname) > 0 && strlen($firstname) < 50)){
    array_push($errors, "Il nome deve essere lungo tra gli 1 e 50 caratteri");
}
if(!(strlen($lastname) > 0 && strlen($lastname) < 50)){
    array_push($errors, "Il cognome deve essere lungo tra gli 1 e 50 caratteri");
}
if(!(strlen($username) > 0 && strlen($username) < 50)){
    array_push($errors, "Lo username deve essere lungo tra gli 1 e 50 caratteri");
}
if(!(strlen($email) > 0 && strlen($email) < 50)){
    array_push($errors, "L'email deve essere formattata nel seguente modo:
indirizzo@dominio.xx");
}
if(!(strlen($password) >= 8)){
    array_push($errors, "La password deve essere almeno lunga 8 caratteri");
}
```

Se anche qui tutto risulta corretto si passa a controllare che le due password siano uguali e che l'email non sia già in uso.

```
if($password == $repassword) {
    $db = (new db_connection)->getUsers();

    foreach ($db as $row) {
        if ($row['email'] == $email) {
            array_push($errors, "L'email è già in uso");
            $exists = true;
        }
    }
```

Se tutto risulta corretto si procede con l'inserire l'utente nel database, questo viene fatto richiamando la seguente funzione.

```
public function addUser($firstname, $lastname, $username, $email, $password){
        $db = $this->getConnection();
        $query = $db->prepare('insert into utente(email, nome, cognome, username,
password, 'nome_ruolo') values (":email", ":firstname", ":lastname", ":username",
":password", ":nome_ruolo")');

        $password = hash('sha256', $password);
        $type = 'utente';

        $query->bindParam(':email', $email, PDO::PARAM_STR);
        $query->bindParam(':firstname', $firstname, PDO::PARAM_STR);
        $query->bindParam(':lastname', $lastname, PDO::PARAM_STR);
        $query->bindParam(':username', $username, PDO::PARAM_STR);
        $query->bindParam(':password', $password, PDO::PARAM_STR);
        $query->bindParam(':nome_ruolo', $type, PDO::PARAM_STR);

        $query→execute();
}
```

Essa esegue l'insert nella tabella utilizzando i prepared statements che impediscono le SQL Inljections.

Se nelle verifiche si riscontrano errori si dovrà rimandare l'utente alla pagina di login e mostrare i messaggi d'errore.

Problemi riscontrati e soluzioni adottate

Oggi ho avuto un solo grande problema che è stato il fatto che, dopo aver inserito l'utente nel database, richiamando la pagina di login lo stile non venisse caricato. Questo era dovuto al fatto che il require (che utilizzavo io) non ricarica la pagina ma aggiunge alla corrente il contenuto di un file php e quindi non si caricavano gli stili. Ho risolto il problema chiedendo aiuto al docente Sartori che mi ha consigliato di utilizzare il metodo header al posto di require.

Punto della situazione rispetto alla pianificazione

Sono in orario.

Programma di massima per la prossima giornata di lavoro

La prossima giornata voglio terminare l’implementazione della pagina di registrazione e iniziare la pagina di login (backend).