Diario di lavoro

Luogo	Balerna
Data	15.05.2020

Lavori svolti

Nella giornata di oggi mi sono occupato di aggiornare la documentazione ed aggiungere i seguenti capitoli:

- Autenticazione
  - Gestione delle sessioni
  - Gestione dei permessi
  - Gestione percorsi e accessi
- Invio di e-mail

Come pianificato mi sono occupato di implementare la funzionalità di recupero password ed ho iniziato la gestione utenti. Per l'implementazione della funzionalità di recupero password come prima cosa ho creato una classe Mail prendendo spunto da un vecchio progetto la quale mi permetterà di inviare messaggi di posta elettronica in un modo semplificato. Il metodo principale della classe Mail è send il quale permette di inviare e-mail:

```php
public static function send($to, $subject, $message)
{
    $eol = "\r\n";

    // header principale
    $headers = "From: <" . self::$fromEmail . ">" . $eol;
    $headers .= "Content-Type: text/html; charset=UTF-8" . $eol;

    return mail($to, $subject, $message, $headers);
}
```

Successivamente ho creato una classe Model chiamata Users che andrà a gestire l'interrogazione con il database per quanto riguarda la tabella user. In questa classe sono presenti svariati metodi i quali permettono di ricavare informazioni da parte della tabella user, ad esempio controllare se un utente esiste attraverso la sua e-mail:

```php
public static function getByEmail($email)
{
    $pdo = Database::getConnection();
    $query = "SELECT * FROM user WHERE email = :email";
    $stm = $pdo->prepare($query);
    $stm->bindParam(':email', $email);
    try {
        $stm->execute();
        return $stm->fetch(\PDO::FETCH_ASSOC);
    } catch (\PDOException $e) {
        return false;
    }
```

```
}
```

Sono inoltre presenti metodi per inserimento, aggiornamento ed eliminazione. Per implementare il recupero e cambio della password andrò ad utilizzare dei token salvati all'interno del database. Per questi token ho creato dunque una classe Model dedicata chiamata Tokens. Questa classe contiene tutti i metodi utilizzati per la generazione, invio ed utilizzo di un token. Ad esempio il metodo di invio per il recupero password è il seguente:

```
public static function sendResetPasswordToken($email)
{
    if (Users::getByEmail($email)) {
        self::deleteToken($email);
        $token = bin2hex(random_bytes(20));
        $hash = hash("sha256", $token);
        try {
            $pdo = Database::getConnection();
            $query = "INSERT INTO token(email, token) VALUE(:email, :token)";
            $stm = $pdo->prepare($query);
            $stm->bindParam('email', $email);
            $stm->bindParam('token', $hash);
            $link = "http://" . $_SERVER['SERVER_NAME'] . "/login/$token";
            $content = "Contenuto";
            return $stm->execute() && Mail::send($email, "Titolo", $content);
        } catch (\PDOException $e) {
        }
    }
    return true;
}
```

Il codice in questione si occupa di recuperare l'utente corrente, eliminare tutti i token se presenti, creare un token in modo randomico, lo salva nel database ed invia per emil. La parte importante di questo codice è il fatto che all'interno del database viene salvato solamente una hash del token in modo tale che se un utente avesse accesso in lettura al database non abbia i mezzi per resettare le password di altri utenti richiedendo un semplice recupero password. All'utente verrà invece inviato il token originale, il concetto è simile a ciò che si usa per salvare e validare le password di accesso. Un altro metodo importante di questa classe è il metodo useToken il quale esegue i controlli sul token passato dall'utente e lo utilizza.

```
public static function useToken($token)
{
    $token = hash("sha256", $token);
    $pdo = Database::getConnection();
    $query = "SELECT email FROM token WHERE token = :token AND CURRENT_TIMESTAMP() -
 created_at <= " . (self::EXPIRE_AFTER * 60);
    try {
        $stm = $pdo->prepare($query);
        $stm->bindParam('token', $token);
        $stm->execute();
        $email = $stm->fetchColumn();
        if ($email) {
            $stm = $pdo->prepare("DELETE FROM token WHERE token = :token");
```

```
            $stm->bindParam('token', $token);
            $stm->execute();
            return $email;
        }
    } catch (\PDOException $e) {
    }
    return false;
}
```

Il metodo in questione si occupa di controllare la presenza del token nel database, di eliminarlo e di ritornare l'utente assegnato ad esso in modo da poter eseguire future azioni su quell'utente. Questa funzionalità viene utilizzata per il cambio password nella classe Model Users:

```
public static function changePassword($token, $password)
{
    if (Validator::isValidPassword($password)) {
        $email = Tokens::useToken($token);
        if ($email) {
            $pdo = Database::getConnection();
            $query = "UPDATE user SET password = :password WHERE email = :email";
            $stm = $pdo->prepare($query);
            $stm->bindParam(':email', $email);
            $stm->bindParam(':password', password_hash($password, PASSWORD_DEFAULT));
            try {
                $_SESSION["login_email"] = $email;
                return $stm->execute();
            } catch (\PDOException $e) {
            }
        }
    }
    return false;
}
```

Il metodo in questione viene utilizzato per eseguire il cambio password sfruttando un token inviato tramite posta elettronica. Essendo che alla creazione di un utente dovranno essere inviate delle credenziali ad un utente per accedere, verrà inviato un link con un token di attivazione che al primo accesso permetterà il cambio password.

Problemi riscontrati e soluzioni adottate

Nessun problema riscontrato.

Punto della situazione rispetto alla pianificazione

Al momento ho terminato la funzionalità di recupero password e cambio password. Ho dunque un leggero vantaggio rispetto alla pianificazione preventiva.

Programma di massima per la prossima giornata di lavoro

Continuare e terminare la gestione degli utenti del sito web.