5. Rejestracja i logowanie

5.1. Opis rozdziału

W tym rozdziale przedstawiono proces stworzenia formularzy rejestracji i logowania oraz oprogramowanie możliwości zarejestrowania nowego użytkownika oraz możliwość jego zalogowania czy też wylogowania z gry.

5.2. Formularz rejestracji

Na samym początku musimy dać użytkownikowi możliwość założenia przez niego nowego konta w grze, użyjemy do tego formularza rejestracji nowego użytkownika. Zaczniemy od dodania nowej akcji rejestracji do kontrolera `home`.

Otwieramy plik kontrolera `home` i dodajemy do niego nową akcję `rejestracjaAction()` wraz z domyślnym szablonem operacji jak przedstawiono poniżej:

Kod 5.2.1. Szablon akcji rejestracji `rejestracjaAction()`

```
public function rejestracjaAction(Engine_UrlParser $parser) {

        $tpl = new Views_Home();
        $mdl = new Models_Home();

        $tpl -> testTopMenu();

        //kod akcji

        $tpl -> view();

}
```

Żeby przejść do strony akcji w pasku adresu wpisujemy *adres_strony.pl/home/rejestracja* i powinna się ukazać pusta strona do której dodamy formularz rejestracji, dobrze by było aby po kliknięciu na przycisk Rejestracja w menu górnym uruchamiała się strona rejestracji. Dodajmy do pliku *engine.php* w katalogu *_views* funkcję, która będzie generowała nasze menu górne.

Kod 5.2.2. Funkcja generująca menu górne

```
public function TopMenu() {

        $this -> setTopMenu('
                <ul>
                        <li><a href="'.APP_HOST.'/home">Strona główna</a></li>
                        <li><a href="'.APP_HOST.'/home/rejestracja">Rejestracja</a></li>
                        <li><a href="'.APP_HOST.'/home/owu">OWU</a></li>
                        <li><a href="'.APP_HOST.'/home/kontakt">Kontakt</a></li>
                </ul>
        ');

}
```

Następnie w funkcji `view()` wywołujemy tą funkcję przed przypisaniem wartości zmiennej `$top_menu`.

Kod 5.2.3. Wywołanie funkcji tworzącej menu górne

```
$this->TopMenu();
$top_menu = $this->getTopMenu();
```

Teraz poprzez kliknięcie odnośnika Rejestracja na górnym menu zostaniemy odesłani do podstrony rejestracji.

Zanim przystąpimy do pisania kodu formularzu musimy jeszcze dodać nową funkcję w dziedziczonym pliku widoków czyli w pliku *engine.php* w katalogu *_views*. Dodajmy funkcję, która będzie dodawała nowy artykuł do strony. Nazwijmy ją `setArticle()` i przekażmy do niej 3 argumenty `$header`, `$section` oraz `$footer` dzięki czemu otrzymamy funkcję szablon do artykułu w którym będziemy mogli ustawić własny nagłówek, stopkę oraz jego treść (**Kod 5.2.4.**).

Kod 5.2.4. Funkcja rysująca artykuł

```
public function setArticle($header, $section, $footer) {
        $this->setContainer('
                <article>
                        <header>'.$header.'</header>
                        <section>'.
                                $section
                        .'</section>
                        <footer>'.$footer.'</footer>
                <article>
        ');
}
```

Kiedy już mamy uruchomioną akcję rejestracji możemy dodać do niej formularz rejestracji zatem w pliku widoku `home` dodajmy nową funkcję z opisem formularza rejestracji (**Kod 5.2.5.**).

Kod 5.2.5. Funkcja opisująca formularz rejestracji

```
public function registerForm($data) {

	$login = (!empty($data[0])) ? $data[0] : null;
	$email = (!empty($data[1])) ? $data[1] : null;

	$this -> setArticle('Formularz rejestracji', '
		<form action="" method="post">
			Login: <input type="text" name="login" value="'.$login.'"><br />
			E-mail: <input type="text" name="email" value="'.$email.'"><br />
			Hasło: <input type="password" name="haslo1"><br />
			Powtórz hasło: <input type="password" name="haslo2"><br />
			<input type="text" name="wyslano" value="1" hidden="hidden"/>
			<input type="submit" value="Zarejestruj">
		</form>
	', '');

}
```

Do funkcji `registerForm()` przekazujemy jeden tablicowy argument `$data`, który będzie nam potrzebny do uzupełnienia formularza gdy podamy złe dane lub wystąpi błąd, aby nie wypełniać wszystkich pól od nowa. Następnie sprawdzamy czy określono poszczególne pola tablicy i przypisujemy ich wartości do zmiennych pomocniczych, które następnie dopisujemy do wartości odpowiednich pól formularza. Pole z hasłem zawsze będzie czyszczone w razie niepowodzenia rejestracji aby zapobiec próbie przechwycenia oraz wymagane będzie dwukrotne podanie hasła aby wykluczyć możliwość pomyłki. Formularz przesyłamy do obecnie uruchomionej podstrony poprzez nie podanie adresu parametrowi `action` formularza. Wszystko przesyłamy metodą POST aby nasze hasło nie zostało wyświetlone w pasku adresu co by naraziło nas na możliwość odczytania hasła. Przed dodaniem przycisku wysyłającego formularz dodano pole `<input type="text" name="wyslano" value="1" hidden="hidden"/>`, będzie to pole określające nam czy formularz został wysłany czy też nie. Pole `wyslano` jest ukryte, bo nie chcemy aby ktoś modyfikował jego wartość domyślną oraz aby niepotrzebnie nie pokazywało się na stronie pole to posiada wartość `1`. Gdy formularz jest już gotowy możemy wywołać jego funkcję w kontrolerze.

Kod 5.2.6. Wywołanie funkcji rysującej formularz

```
$pokazFormularz = true;
$wyslano = $parser -> getArg('wyslano');
$login = $parser -> getArg('login');
$email = $parser -> getArg('email');
$haslo1 = $parser -> getArg('haslo1');
$haslo2 = $parser -> getArg('haslo2');

//operacje wykonane po wysłaniu formularza

if($pokazFormularz) $tpl -> registerForm(array(
	$login,
	$email
));
```

Najpierw tworzymy zmienną pomocniczą `$pokazFormularz`, która będzie użyta do sprawdzenie czy formularz ma zostać wyświetlony czy też nie. Następnie odczytujemy potencjalne wartości przesłane z formularza, a następnie sprawdzamy czy formularz ma zostać wyświetlony i wyświetlamy go. Jako argument funkcji przekazujemy tablicę z odczytaną wartością z pola login i e-mail.

WAŻNE: Nie przekazujemy do formularza odczytanego hasła gdyż, ktoś niepożądany mógłby je odczytać!

Możemy już przetestować czy formularz jest wyświetlany poprawnie (**Rys. 5.2.1.**). Jeśli wszystko jest w porządku to możemy przejść do sprawdzenie czy wysłano formularz oraz zarejestrowania użytkownika.

Rys. 5.2.1. Formularz rejestracji nowego użytkownika

Zanim jednak przejdziemy do sprawdzania czy formularz został wysłany zmodyfikujmy funkcje wyświetlające komunikaty o błędach, powodzeniach i informacjach. Przejdźmy do pliku *engine.php* w katalogu *_views* i edytujmy funkcje `error`, `success` oraz `info` (**Kod 5.2.7.**). Dzięki temu komunikaty będą wyświetlane w okienku jak np. formularz logowania. Po zapisaniu zmian możemy przejść do sprawdzania czy wysłano formularz.

Kod 5.2.7. Modyfikacja funkcji komunikatów

```
public function error($com) {
        $this -> setArticle('BŁĄD', '<div id="com" class="error">'.$com.'</div>', '');
}

public function success($com) {
        $this -> setArticle('SUKCES', '<div id="com" class="success">'.$com.'</div>', '');
}

public function info($com) {
        $this -> setArticle('INFORMACJA', '<div id="com" class="info">'.$com.'</div>', '');
}
```

W miejsce komentarza `//operacje wykonane po wysłaniu formularza` w pliku widoku `home` wstawimy skrypt, który najpierw sprawdzi czy wysłano formularz i jeśli tak to sprawdzi czy wszystkie jego pola zostały wypełnione.

Kod 5.2.8. Sprawdzenie czy wysłano i wypełniono formularz rejestracji

```
if($wyslano) {
        if($login != '' && $email != '' && $haslo1 != '' && $haslo2 != '') {
                //dodanie nowego uzytkownika do bazy danych
        } else $tpl -> error('Nie podano wszystkich danych!');
}
```

Pierwsza instrukcja warunkowa sprawdza czy wartość zmiennej `$wyslano` jest prawdziwa czyli nie pusta lub różna od ZERA. Jeśli zmienna ta będzie posiadała wartość 1 przejdziemy do następnej instrukcji warunkowej aby sprawdzić czy wypełniono wszystkie pola formularza. Jeśli wszystkie pola formularza zostały wypełnione możemy przekazać je do funkcji, która sprawczy poprawność danych i doda nowego użytkownika do bazy danych. Jeśli z kolei którekolwiek pole formularza zostanie nie wypełnione zostanie wyświetlony komunikat o błędzie (**Rys. 5.2.2.**).

Rys. 5.2.2. Komunikat o błędzie w przypadku nie wypełnienia wszystkich pól formularza.

W przypadku kiedy uruchomimy stronę rejestracji bez kliknięcia przycisku „Zarejestruj” nie powinien się nam pokazać, żaden komunikat i tak się dzieje dzięki użyciu pola ukrytego w formularzu.

Teraz skoro wszystkie dane są już odbierane od formularza możemy je przekazać do funkcji, która się nimi zajmie pod kątem ich poprawności oraz doda poprawne dane do bazy danych. Najpierw dodajmy do klasy `Models_Engine` dwa pola:

- `$error_code` – pole to będzie przechowywać numer błędu powstałego podczas wykonywania funkcji, numer błędu może być dowolny i dla każdej funkcji mogą wystąpić jednakowe numery błędów. Numer błędu 0 rezerwujemy dla braku błędu. Każdy inny numer określa jakiś błąd, który jest opisywany w kontrolerze i wyświetlany na ekranie.
- `$result` – pole to będzie przechowywać wynik działania funkcji jeśli takowy powstanie

Kod 5.2.9. Dodanie pól do klasy `Models_Engine`

```
public $pdo,
       $error_code,
       $result;
```

Dodajmy więc nową funkcję `insertNewUser` do modelu `home`, której zadaniem będzie sprawdzenie poprawności przekazanych jej danych z formularza oraz dodanie nowego użytkownika.

Kod 5.2.10. Funkcja sprawdzająca poprawność danych rejestracji i dodająca nowego użytkownika do bazy danych

```
public function  insertNewUser($data) {
        $pdo = $this->pdo;
        $this -> error_code = 0;

        $login = $data[0];
        $email = $data[1];
        $haslo1 = $data[2];
        $haslo2 = $data[3];

        if(filter_var($email, FILTER_VALIDATE_EMAIL)) {

                if($haslo1 === $haslo2) {

                        //dodanie nowego konta do bazy danych

                } else $this -> error_code = -2; //podane hasla sa rozne

        } else $this -> error_code = -1; //podano niepoprawny email
}
```

W razie wystąpienia błędu zostanie on zapisany do pola `$error_code` klasy. Musimy pamiętać o wyzerowaniu pola `$error_code` na początku każdej funkcji aby po jej wykonaniu nie odczytać numeru błędu z poprzednio wykonanej funkcji. Funkcja `filter_var` posłuży do sprawdzenia poprawności adresu e-mial. Dobrze jest komentować błędy zwracane przez funkcję aby nie było potem problemów z tym co konkretny błąd oznacza.

Jeśli sprawdzenie poprawności już mamy za sobą należy zająć się bazą danych. Najpierw należy sprawdzić czy nie zarejestrowano już użytkownika o danej nazwie login lub adresie e-mail. Następnie można dodać nowe konto użytkownika do bazy danych (**Kod 5.2.11.**).

Kod 5.2.11. Dodanie konta nowego użytkownika do bazy danych

```
$select = $pdo -> prepare("SELECT * FROM `konto` WHERE `login` = :login OR `email` = :email");
$select -> bindValue(':login', $login, PDO::PARAM_STR);
$select -> bindValue(':email', $email, PDO::PARAM_STR);
$select -> execute();

if(!$select -> rowCount()) {

	$insert = $pdo -> prepare("

		INSERT INTO
		`konto` (`login`, `haslo`, `email`, `aktywny`, `data`)
		VALUES (:login, :haslo, :email, :aktywny, :data)

	");

	$insert -> bindValue(':login', $login, PDO::PARAM_STR);
	$insert -> bindValue(':haslo', sha1($haslo1), PDO::PARAM_STR);
	$insert -> bindValue(':email', $email, PDO::PARAM_STR);
	$insert -> bindValue(':aktywny', 1, PDO::PARAM_INT);
	$insert -> bindValue(':data', date('Y-m-d H:i:s'), PDO::PARAM_STR);

	if(!$insert -> execute()) $this -> error_code = -4; //proba doadnia nowego konta nie
powiodla sie

} else $this -> error_code = -3; //odczytano uzytkownika o takim loginie lub adresie email
```

Jeśli mamy już gotowy skrypt dodający nowe konto użytkownika należy uruchomić go w kontrolerze oraz sprawdzić błędy jakie zwraca i wyświetli stosowne komunikaty. W kontrolerze `home` w miejsce komentarza `//dodanie nowego uzytkownika do bazy danych` dodajmy kod, który wywoła funkcję powyżej napisaną. (**Kod 5.2.12.**).

Kod 5.2.12. Wywołanie funkcji `insertNewUser` dodającej nowego użytkownika

```
$mdl -> insertNewUser(array(

	$login,
	$email,
	$haslo1,
	$haslo2

));

//sprawdzenie błędów
```

Do funkcji jako parametr wstawiamy tablicę z danymi odczytanymi z formularza. Teraz po wysłaniu formularza nowy użytkownik zostanie lub nie zostanie dodany do bazy danych. Następnym krokiem jest sprawdzenie numerów błędów, najwygodniej będzie tutaj użyć instrukcji `switch()`.

Kod 5.2.13. Sprawdzenie numerów błędów oraz wyświetlenie stosownych komunikatów

```
switch($mdl -> error_code) {

	case 0 :
		$tpl -> success('Nowe konto użytkownika zostało zarejestrowane. Możesz się teraz zalogować.');
		$pokazFormularz = false;
		break;

	case -1 :
		$tpl -> error('Podany adres email nie posiada poprawnej konstrukcji!');
		break;

	case -2 :
		$tpl -> error('Podane hasłą nie są jednakowe!');
		break;

	case -3 :
		$tpl -> error('Zarejestrowano już użytkownika o podanej nazwie login lub adresie e-mail!');
		break;

	case -4 :
		$tpl -> error('Próba dodania nowego konta nie powiodła się!');
		break;

}
```

Numer błędu ZERO informuje nas, że nie ma błędów i utworzono nowe konto użytkownika więc wyświetlamy odpowiedni komunikat i nie pozwalamy na ponowne pokazanie się formularza. W pozostałych przypadkach odpowiednio opisujemy komunikaty, aby użytkownik wiedział co się stało i nie blokujmy wyświetlania formularza.

Dobrze jest zablokować możliwość rejestracji dla zalogowanych użytkowników dodając instrukcję warunkową, która sprawdzi czy sesja została ustawiona dla zalogowanego użytkownika.

Kod 5.2.14. Zabezpieczenie przez rejestracją zalogowanego użytkownika

```
public function rejestracjaAction(Engine_UrlParser $parser) {

	$tpl = new Views_Home();
	$mdl = new Models_Home();

	if(!isset($_SESSION['login'])) {

		//skrypt rejestracji

	} else $tpl -> error("Nie możesz korzystać z formularza logowania będąc zalogowanym!");

	$tpl -> view();

}
```

To by było na tyle jeśli chodzi o rejestrowanie nowych kont użytkowników.

5.3. Logowanie

Mamy już zarejestrowanych użytkowników ale nadal nie mają oni możliwości zagrania w naszą grę, bo brakuje nam formularza logowania. Zacznijmy więc od stworzenia szablonu formularza logowania w widoku `engine`.

Kod 5.3.1. Formularz logowania użytkownika

```
public function loginForm() {

        $this -> setAside('

                <div id="loginForm">
                        <div id="naglowek">
                                Formularz logowania
                        </div>
                        <div id="formularz">
                                <form action="'.APP_HOST.'/home/zaloguj" method="post">

                                        Login:<br />
                                        <input type="text" name="login" /><br />
                                        Hasło:<br />
                                        <input type="password" name ="haslo" /><br />
                                        <input type="submit" value="Zaloguj się" />

                                </form>
                        </div>
                </div>

        ');

}
```

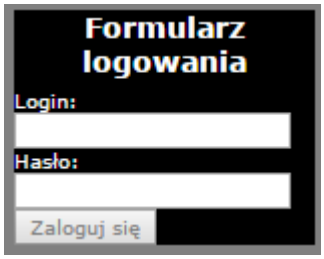
Formularz logowania będzie wyświetlane w lewym panelu tylko jeśli nie będzie zalogowany użytkownik. W formularzu podajemy tylko login i hasło do konta użytkownika. Proces logowania zostaje dokończony po przekierowaniu na stronę logowania. Aby teraz wywołać funkcję rysującą formularz logowania należy dodać do konstruktora widoku wywołanie tej funkcji poprzedzone instrukcją warunkową, której zadaniem będzie sprawdzenie czy wyświetlić formularz sprawdzając czy zalogowano użytkownika.

Kod 5.3.2. Zmodyfikowany konstruktor klasy `Views_Engine` poprzez dodanie wywołania formularza

```
public function __construct($http = 'text/html; charset=UTF-8') {
        $this->setTitleTag('Krasnale');
        $this->setMetaTagHttp('Content-Type', $http);
        $this->setMetaTagLink('stylesheet', '/_views/style.css', 'text/css', '');
        $this->setFooter('&copy <a href="home">Podziemne królestwo</a>');

        if(!isset($_SESSION['login'])) $this -> loginForm(); //wywołanie formularza logowania
}
```

Teraz można sprawdzić jak prezentuje się formularz logowania na stronie (**Rys. 5.3.1.**). Można tutaj zająć się edycją kodu CSS, ale na tym etapie prac odpuścimy to sobie i zostawmy jako ćwiczenie dla chętnych. Przejdźmy zatem do skryptu logującego użytkownika.



Rys. 5.3.1. Formularz logowania użytkownika

Skoro mamy już formularz logowanie to trzeba teraz przejąć dane wysłane z formularza i sprawdzić czy użytkownik o podanym loginie w ogóle istnieje w bazie *kont* oraz czy podał poprawne hasło. Zaczniemy od utworzenia nowej akcji `zalogujAction` w kontrolerze `home`.

Kod 5.3.3. Akcja `zalogujAction` odpowiedzialna za logowanie użytkownika

```
public function zalogujAction(Engine_UrlParser $parser) {

        $tpl = new Views_Home();
        $mdl = new Models_Home();

        if(!isset($_SESSION['login'])) {

                //odczytanie wartości wysłanych z formularza

                //funkcja logująca

                //przekierowanie do strony głównej

        } else $tpl -> error("Nie możesz korzystać z opcji logowanie będąc zalogowanym!");

        $tpl -> view();

}
```

Podobnie jak przy rejestracji na początku sprawdzamy czy użytkownik już się nie zalogował poprzez sprawdzenie odpowiedniej zmiennej sesyjnej. W tym przypadku zmienną sesyjną przetrzymującą identyfikator zalogowanego użytkownika będzie zmienna `$_SESSION['login']`. Skoro nie mamy zalogowanego jeszcze użytkownika musimy mu to umożliwić. Najpierw odczytajmy dane wysłane przez formularz(**Kod 5.3.4.**).

Kod 5.3.4. Odczytanie danych logowania wysłanych z formularza

```
$login = $parser -> getArg('login');
$haslo = $parser -> getArg('haslo');

if($login != '' && $haslo != '') {

	//funkcja logująca

	//przekierowanie do strony głównej

} else $tpl -> error('Nie podano wszystkich danych logowania!');
```

W przypadku tego formularza nie musimy dodawać pola kontrolnego, które sprawdzi czy wysłano dane z formularza jak to miało miejsce w przypadku rejestracji. Kod powyżej najpierw pobiera potencjalne dane wysłane przez formularz, a następnie sprawdza czy zostały wypełnione oba pola formularza. Jeśli nie wypełniono obydwóch pól formularza zostanie wyświetlony odpowiedni komunikat. Jeśli z kolei podano obydwie dane należy uruchomić funkcję, która przetworzy dane i sprawdzi czy istnieje dany użytkownik oraz czy podał poprane hasło do swojego konta.

Zaczniemy od napisania funkcji, która będzie odpowiedzialna za logowanie użytkownika. Funkcja ta będzie posiadała nazwę `userLogin()` i umieścimy ją w modelu `home`.

Kod 5.3.5. Funkcja logowania użytkowników

```
public function loginUser($data) {

	$pdo = $this -> pdo;
	$this -> error_code = 0;

	$login = $data[0];
	$haslo = $data[1];

	$select = $pdo -> prepare("SELECT * FROM `konto` WHERE `login` = :login");
	$select -> bindValue(':login', $login, PDO::PARAM_STR);
	$select -> execute();

	if($select -> rowCount()) {

		$row = $select -> fetch(PDO::FETCH_ASSOC);

		if($row['haslo'] === sha1($haslo)) {

			$_SESSION['login'] = $row['id'];

		} else $this -> error_code = -2; //podano nipoprawne haslo dla tego uzytkwonika

	} else $this -> error_code = -1; //nie odczytano uzytkownika o danym loginie

}
```

Na początku zerujemy kod błędu i dla przejrzystości odczytujemy dane z przekazanej tablicy. Następnie pobieramy z bazy danych z tablicy *konto* rekord powiązany z loginem podanym

w formularzu logowania. Instrukcją warunkową sprawdzamy czy wykonanie zapytania zwróciło jakiekolwiek rekordy. Jeśli mamy zwrócony rekord zapytania to wiemy, że istnieje konto użytkownika o loginie podanym w formularzu logowania. Następnie instrukcją warunkową porównujemy hasło odczytane z bazy danych dla użytkownika z zakodowanym hasłem podanym w formularzu logowania. Ostatecznie jeśli wszystkie podane dane są poprawne to przypisujemy do zmiennej sesyjnej `$_SESSION['login']` numer identyfikatora konta użytkownika z bazy danych.

Teraz należy wywołać funkcję logującą w konstruktorze. Jako jej parametry wstawimy tablicę z danymi wysłanymi z formularza.

Kod 5.3.6. Wywołanie funkcji logującej

```
$mdl -> loginUser(array(

        $login,
        $haslo

));

//obsługa błędów

//przekierowanie do strony głównej
```

Następnie obsłużmy zmienną z numerem błędu jaki zwróci funkcja logująca.

Kod 5.3.7. Obsługa błędów funkcji logującej

```
switch($mdl -> error_code) {

        case 0 :
                $tpl -> success('Użytkownik zalogowany pomyślnie.');

                //przekierowanie do strony głównej

                break;

        case -1 :
                $tpl -> error("Nie zarejestrowano użytkownika o takim loginie!");
                break;

        case -2 :
                $tpl -> error("Podano złe hasło!");
                break;

}
```

Na końcu musimy przekierować/przeładować stronę do strony głównej aby użytkownik nie widział już w panelu bocznym formularza logowania tylko „Kartę Władcy” i wszystkie kontrolki przystosowane do strony zalogowanego użytkownika. Przekierowanie ma nastąpić w przypadku poprawnego zalogowania się użytkownika. Tak więc należy zmodyfikować trochę komunikat w przypadku zalogowania się użytkownika oraz dodać przekierowanie z opóźnieniem powiedzmy 5 sekundowym.

Kod 5.3.8. Przekierowanie po zalogowaniu użytkownika

```
case 0 :
        $tpl -> success('Użytkownik zalogowany pomyślnie. Za chwilę nastąpi odświeżenie strony.');

        header('Refresh: 5; url='.APP_HOST);

        break;
```

Logowanie już jest gotowe, rejestracja podobnie, a teraz potrzebny jest mechanizm pozwalający wyjść z gry aby nikt inny nie mógł się dobrać do naszego konta.

5.4. Wylogowanie użytkownika

Wylogowanie będzie polegało na dodaniu do panelu bocznego przycisku przekierowującego na stronę wylogowywania, która wyczyści zmienne sesyjne. Przycisk wylogowywania dodamy do konstruktora klasy `Views_Engine`.

Kod. 5.4.1. Modyfikacja konstruktora klasy `Views_Engine` w celu dodania przycisku wylogowania użytkownika

```
if(!isset($_SESSION['login'])) $this -> loginForm(); //wywołanie formularza logowania
else { //opcje widoczne dla zalogowanego użytkownika

        $this -> setAside('

                <a href="'.APP_HOST.'/home/wyloguj">Wyloguj się</a>

        ');

}
```

Teraz po zalogowaniu się w miejscu gdzie znajdował się formularz logowania powinien pokazać się przycisk „Wyloguj się”. Kliknięcie w ten przycisk powinno przekierować nas na stronę, która wyczyści zmienne sesyjne. Tak więc dodajmy do konstruktora `home` nową akcję `wylogujAction()`.

Kod. 5.4.2. Akcja `wylogujAction` wylogowująca użytkownika

```
public function wylogujAction(Engine_UrlParser $parser) {

        $tpl = new Views_Home();
        $mdl = new Models_Home();

        if(isset($_SESSION['login'])) {

                //funkcja wylogowywania

        } else $tpl -> error("Nie możesz się wylogować skoro nie jesteś zalogowany!");

        $tpl -> view();

}
```

Dodajmy teraz do modelu **`home`** nową metodę `logoutUser()`, która wyczyści zmienną sesyjną.

Kod. 5.4.3. Funkcja czyszcząca zmienną sesyjną z identyfikatorem zalogowanego użytkownika

```
public function logoutUser() {

        $this -> error_code = 0;

        unset($_SESSION['login']);

        if(isset($_SESSION['login'])) $this -> error_code = -1; //nie udalo sie wyczyscic sesji

}
```

Po wyczyszczeniu zmiennej należy sprawdzić czy oby na pewno zmienna jest pusta. Jeśli tak to kończymy funkcję z numerem błędu 0 w przeciwnym wypadku -1 czyli nie udało się wyczyścić zmiennej sesyjnej przechowującej identyfikator konta użytkownika.

Jeśli funkcja jest gotowa możemy ją wywołać i sprawdzić numer błędu i wyświetlić odpowiedni komunikat oraz w przypadku pomyślnego wylogowania należy odświeżyć stronę aby uniknąć zdarzenia kiedy użytkownik jest wylogowany, a mimo to pokazany jest jego „Karta Władcy”.

Kod. 5.4.4. Dodanie wywołanie funkcji wylogowującej oraz sprawdzenie numeru błędu

```
$mdl -> logoutUser();

switch($mdl -> error_code) {

	case 0 :
		$tpl -> success('Użytkownik został wylogowany pomyślnie. Za chwilę nastąpi
odświeżenie strony.');

		header('Refresh: 5; url='.APP_HOST);

		break;

	case -1 :
		$tpl -> error('Próba wylogowania użytkownika nie powiodła się!');
		break;

}
```

I to by było na wszystko w tym rozdziale. Teraz mamy możliwość na rejestrowanie nowych użytkowników, logowanie się zarejestrowanych oraz wylogowanie się zalogowanych. Wynik prac opisanych w tym rozdziale został zamieszczony w pliku „*roz5.rar*”.